Méthodes de sélection de collections dans un environnement de recherche d'informations distribuée

Faïza Abbaci

École Nationale Supérieure des Mines de Saint-Etienne

Université Jean Monnet de Saint-Etienne

Université de Neuchâtel

N° d'ordre : 311

THÈSE

Présentée par Faïza ABBACI

pour obtenir le titre de

Docteur *(Spécialité Informatique)*

DE L'UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE et DE L'ÉCOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ETIENNE

et du titre de

Docteur ès Sciences (informatique)
DE L'UNIVERSITÉ DE NEUCHÂTEL

Méthodes de sélection de collections dans un environnement de recherche d'informations distribuée

Composition du jury :

Monsieur	Mohand BOUGHANEM	Président et rapporteur
Monsieur	Jian Yun NIE	Rapporteur
Madame	Christine LARGERON	examinatrice
Monsieur	Pierre Jean ERARD	examinateur
Messieurs	Jean-Jacques GIRARDOT	Directeur de thèse
	Michel BEIGBEDER	Directeur de recherche
	Jacques SAVOY	Co-directeur de thèse

Date et lieu de soutenance : 20 juin 2003 à Saint-Etienne

École Nationale Supérieure des Mines de Saint-Etienne

Université Jean Monnet de Saint-Etienne

Université de Neuchâtel

N° d'ordre : 311

THÈSE

Présentée par Faïza ABBACI

pour obtenir le titre de

Docteur *(Spécialité Informatique)*

DE L'UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE et DE L'ÉCOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ETIENNE

et du titre de

Docteur ès Sciences (informatique)
DE L'UNIVERSITÉ DE NEUCHÂTEL

Méthodes de sélection de collections dans un environnement de recherche d'informations distribuée

Composition du jury :

Monsieur	Mohand BOUGHANEM	Président et rapporteur
Monsieur	Jian Yun NIE	Rapporteur
Madame	Christine LARGERON	examinatrice
Monsieur	Pierre Jean ERARD	examinateur
Messieurs	Jean-Jacques GIRARDOT	Directeur de thèse
	Michel BEIGBEDER	Directeur de recherche
	Jacques SAVOY	Co-directeur de thèse

Date et lieu de soutenance : 20 juin 2003 à Saint-Etienne

À mes parents qui ont tellement fait pour moi.

À mon mari Franck et mon fils Anir.

À ma soeur Nabila et mes frères Lyes et Faouzi.

À tout le reste de ma famille et ma belle famille.

À tous ceux que j'aime.

Remerciements

Je remercie d'abord l'équipe RIM qui m'a accueillie et où j'ai pu mener à bien mes travaux de recherche.

Je remercie Michel Beigbeder, mon directeur de recherche, avec qui j'ai travaillé en toute sérénité.

Je tiens à exprimer ma gratitude à Jacques Savoy, d'abord pour la confiance qu'il avait en moi, ensuite pour son accueil dans son laboratoire à Neuchâtel et son formidable encadrement.

Je remercie vivement Mohand Boughanem d'avoir accepté d'être rapporteur de cette thèse. Je le remercie pour sa lecture approfondie de mon document et pour ses commentaires constructifs.

Je remercie vivement Jian Yun Nie d'avoir accepté spontanément d'être rapporteur de ce travail. Je le remercie pour ses nombreuses remarques qui ont permis d'améliorer la qualité de ce rapport. Je tiens à lui exprimer toute ma gratitude de s'être déplacé de Montréal pour assister à ma soutenance.

Je remercie également Christine Largeron d'avoir participé au jury de thèse. Son intérêt pour mes travaux m'honore. Je la remercie d'avoir trouvé le temps de lire avec attention ma thèse.

Je remercie Jean Pierre Erard d'avoir bien voulu faire partie de mon jury et de s'être déplacé de Suisse pour ma soutenance.

Enfin, je remercie tous ceux qui, durant ces années de recherche, m'ont soutenue et accompagnée, notamment mon mari Franck qui a relu ma thèse et qui a fait office de public à mes nombreuses pré-soutenances. Je remercie Yves Rasolofo pour les nombreuses et agréables heures de travail que nous avons passé ensemble.

Table des matières

Chapitre 1

Introduction

Table des matières

Nul ne peut nier les changements que ce nouveau média appelé Internet a apportés dans notre vie quotidienne. Les avantages qu'Internet offre en termes de facilité, de temps, d'argent et même d'organisation sont autant d'éléments qui ont incité à son adoption parmi nous en quelques années.

De plus en plus d'administrations, d'associations, de sociétés, d'entreprises, etc. s'appuient sur Internet à travers leurs applications informatiques. Le développement technologique dans les domaines des transmissions, du stockage, de la sécurité et le développement d'outils pour l'Internet ont permis son extension spectaculaire ces dernières années.

La croissance exponentielle des serveurs et des sites sur Internet a donné naissance à un gigantesque gisement d'information. Pour se retrouver dans cette masse de données, d'innombrables systèmes de recherche d'information ont été installés sur Internet afin de guider l'utilisateur vers l'information recherchée. Ces systèmes ont tous le même but, répondre au mieux à la requête de l'utilisateur et ceci en couvrant au mieux toute l'information disponible en ligne.

Dans la suite de ce rapport, nous nous intéressons particulièrement aux systèmes de recherche d'information sur Internet (SRIs). Ces derniers se divisent en deux catégories selon la stratégie utilisée pour indexer les documents qu'ils couvrent. Dans la première catégorie se trouvent les systèmes de recherche d'information centralisée (SRICs) qui construisent un index unique qui sera consulté à chaque réception d'une requête de l'utilisateur. Ce type de système a des limites, la plus importante étant son extensibilité réduite face à la prolifération spectaculaire des données sur Internet. Dans la deuxième catégorie, on retrouve les systèmes de recherche d'information distribuée (SRIDs) qui viennent apporter des solutions aux limites des SRICs en distribuant les processus d'indexation et de recherche.

Un SRID est composé d'un courtier qui représente le cœur du système. L'utilisateur formule son besoin d'information sous forme de requête et la soumet au courtier. Le courtier choisit un certain nombre de serveurs qu'il juge aptes à répondre d'une façon satisfaisante à la requête. Cette opération est appelée sélection de serveurs. Puis, le courtier transmet la requête aux serveurs ainsi sélectionnés. Les serveurs interrogés transmettent leurs réponses au courtier sous forme de listes de pointeurs vers des documents (URLs dans le cas du Web). Le courtier se charge alors de fusionner ces listes afin de constituer une liste unique de pointeurs qui sera présentée à l'utilisateur.

Dans cette thèse nous nous intéressons aux SRIDs et particulièrement à l'étape de sélection de serveurs. Nous proposons une méthode d'acquisition de données concernant le contenu des serveurs, ces données permettront d'effectuer la sélection. Nous proposons en outre plusieurs méthodes de calcul du score de chaque serveur par rapport à une requête donnée. Enfin, à partir du score associé à chaque serveur, des méthodes de sélection déterminent les serveurs à interroger.

Dans le reste de ce chapitre nous présentons, dans la section 1.1, les SRIs d'une manière générale, le mode de fonctionnement des SRICs, les problèmes posés par ce type de système, et nous présentons également, comment les SRIDs apportent des solutions à ces problèmes. Nous détaillons, dans la section 1.2, la problématique de cette thèse et, dans la section 1.3, les objectifs visés et nos contributions. Nous terminons ce premier chapitre par le plan de ce manuscrit.

1.1 La recherche d'information (RI)

1.1.1 Système de recherche d'information (SRI)

Pour répondre à un besoin d'information, un humain peut s'adresser à des personnes de son entourage ou rechercher l'information sur des supports physiques dans les bibliothèques, les journaux, livres, revues, etc.

Dans le cadre d'un SRI, la recherche d'information (RI) est basée sur quatre entités, à savoir, l'Homme, le besoin d'information, l'information, l'outil d'acquisition d'information. Le besoin d'information est exprimé par l'Homme sous forme d'une **requête**. Suivant le développement technologique, l'apparition des ordinateurs et de l'Internet, les outils développés dans le domaine de la RI ont aussi évolué. Cette évolution touche la manière d'indexer, de stocker et de rechercher l'information contenue dans les documents. Un **document** est un objet qui véhicule des informations pouvant prendre plusieurs formes notamment texte, son, images, vidéo, etc.

Nous nous intéressons exclusivement à la RI automatisée où les documents sont sous forme numérique (ou du moins leur représentation), et pour lesquels l'outil servant à effectuer la recherche est un système informatique. Un Système de recherche d'Information n'est pas un système *Question/Réponse*: il n'est pas sensé répondre explicitement à une requête, mais simplement sur l'existence (ou non) et la localisation de documents ayant rapport avec sa demande [Ris79]. Un SRI est donc un outil informatique qui permet à l'utilisateur d'exprimer son besoin d'information à l'aide d'une requête et qui retrouve les documents pertinents à cette requête parmi l'ensemble des documents qu'il gère, ensemble appelé corpus du système. Très souvent les SRIs retournent des listes de liens vers des documents. Cette liste est triée selon le degré de pertinence (calculé par le SRI), appelé aussi score, de chacun des documents qu'elle contient.

Les problèmes dans le domaine de la RI sont de deux sortes [BY99] :

- des problèmes liés à l'utilisateur : sa capacité à déterminer ses besoins d'information et sa capacité à exprimer ceux-ci par une requête. La conséquence est que cette dernière n'est qu'une description partielle de son besoin d'information ;
- des problèmes liés au système : essentiellement sa capacité à identifier les documents pertinents à la requête.

Les trois notions principales qui se dégagent dans un SRI sont en premier lieu le document, en deuxième lieu la requête — ces deux premiers éléments constituent les données— en troisième lieu le processus de traitement qui établit une

correspondance entre les deux premiers éléments.

1. Le **document :** le document est l'entité minimale qui encapsule l'information. Si cette information correspond au besoin d'information de l'utilisateur (exprimé par une requête), alors la pertinence du document est établie et un lien vers ce document est retourné dans la réponse à la requête.

2. La **requête :** souvent la requête est considérée comme un document dont la taille est très réduite et qui ne satisfait pas nécessairement aux règles de syntaxe habituelle [Kor97]. La requête est généralement sous forme de mots-clés [BY99], dans ce cas, la requête est une suite de termes qui véhiculent la sémantique du besoin d'information.

3. **L'appariement :** l'appariement consiste à associer pour une requête donnée la liste des documents qui lui sont pertinents. Deux paradigmes peuvent se distinguer :

   - **appariement exact** où un document est jugé pertinent s'il vérifie tous les critères spécifiés dans la requête [BY99] et ce jugement est binaire.
   - **appariement avec classement** où un degré de pertinence est attribué à chaque document en fonction de sa similarité sémantique avec la requête. Les documents sont alors ordonnés avant d'être présentés à l'utilisateur.

Nous détaillons dans la section qui suit quelques concepts qui nous paraissent fondamentaux afin de cerner le domaine de la RI.

1.1.1.1 Quelques concepts de base dans le domaine de la RI

1. Un **document :** un document est d'un point de vue fonctionnel défini comme étant une entité atomique qui peut être recherchée, retrouvée, et consultée, sans pour autant être nécessairement physiquement sauvegardée comme une entité unique. D'un point de vue logique, un document est une entité véhiculant une ou plusieurs informations, présentées sous des formes variées (texte, son, image, vidéo, multimédia). Un document peut être un article d'un journal, un livre, un chapitre, une section, un paragraphe, une phrase, une page hypertexte, une image, un fichier. Un même document peut être écrit dans plusieurs langues (comme par exemple, la jurisprudence européenne).

2. Une **requête :** Une requête véhicule le besoin d'information de l'utilisateur. elle contient en général un ensemble de mots clés éventuellement connectés par des opérateurs booléens. On la rencontre également sous

forme d'une phrase ou d'un paragraphe. Elle représente la description des spécifications des documents souhaités. Mais le plus souvent cette description est courte et ambiguë et ne spécifie pas tous les détails du besoin d'information. La requête est exprimée dans un langage d'interrogation. Les langages d'interrogation sont divers, les serveurs peuvent supporter par exemple :

- des requêtes simples de mots clés ;
- des requêtes booléennes où les mots clés sont liés par des opérateurs booléens (AND, OR, NOT) ;
- des requêtes structurées basées sur des attributs tels que les noms d'auteurs, la date de parution etc. ;
- des requêtes sous forme d'expressions régulières ;
- des requêtes complexes qui englobent les types précédents.

3. Un **représentant logique d'un document :** Le plus souvent les SRI n'utilisent pas directement les documents dans leurs processus de traitement, mais plutôt les représentants de ceux-ci.
Le représentant d'un document est une description brève de son contenu. Cette description dépend de l'algorithme d'indexation et d'appariement utilisé. Employer un représentant plutôt que l'original d'un document n'est pas sans impact sur la qualité de la recherche. En effet, ceci provoque une perte d'information [Kor97].
Une façon usuelle de représenter les documents se base sur l'idée que la sémantique d'un document ou d'une requête peut être exprimée par un ensemble de mots-clés [BY99], le problème est alors de choisir les mots-clés. Cette opération peut être effectuée manuellement ou automatiquement. Dans ce dernier cas, la méthode la plus simple consiste à segmenter le document en mots, mais cette méthode est coûteuse, et génère beaucoup de bruit (défini plus bas) [BY99]. En effet, les mots d'un document n'ont pas tous la même capacité pour décrire le contenu du document. Pour y remédier, les systèmes peuvent choisir d'appliquer une ou plusieurs des opérations suivantes :

   (a) supprimer les mots vides (mots fonctionnels) tels que les articles, les pronoms, les conjonctions, les prépositions. Ces mots sont regroupés dans une liste appelée anti-dictionnaire (en anglais stoplist). Les mots vides ne sont pas des mots-clés efficaces, car les utilisateurs ne s'en servent pas pour effectuer la recherche. L'élimination des mots vides réduit la taille du représentant d'un document, ce qui réduit le coût en espace de la segmentation en mots. En outre, cette opération permet de réduire le bruit dans la réponse car les mots vides existent dans la majorité des documents ;

   (b) appliquer la lemmatisation, qui consiste à remplacer toutes les formes morphologiques d'un même mot avec son lemme. Par exemple, on souhaite remplacer les termes "juger", "jugement", "juge", "jugeons"

par "juger" . Cette méthode permet également de réduire la taille du représentant d'un document et d'améliorer la réponse. En effet, ne pas considérer la différence entre, par exemple, deux formes conjuguées d'un même verbe, peut aider à sélectionner des documents pertinents qui ne l'auraient pas été si cette différence avait été prise en compte ;

(c) sélectionner les termes qui représentent le mieux le contenu sémantique du document et leur attribuer un poids indiquant leur importance dans le document. Plusieurs critères de sélection peuvent être considérés, notamment :

- la fréquence d'occurrence des termes : on retient les termes dont la fréquence d'occurrence dépasse un seuil prédéfini. Dans ce cas, plus un terme est fréquent, plus sa pondération doit augmenter dans la représentation d'un document ;
- le degré de discrimination des termes : un terme apparaissant dans beaucoup de documents n'a pas un pouvoir discriminatoire, il ne permet pas de distinguer les documents pertinents. Ainsi, si un terme apparaît dans quelques documents du corpus du système, sa pondération doit être plus forte qu'un terme apparaissant dans beaucoup de documents de ce corpus.

Les deux critères précédents peuvent être conjointement pris en compte par la pondération habituelle $tf_{ik} * idf_k$. Les deux valeurs tf_{ik} et idf_k font référence à la fréquence du terme t_k dans le document d_i et du pouvoir discriminant de t_k. Cette dernière valeur est calculée selon la formule de Sparck Jones [Jon72] :

$$idf_k = \log_2(\frac{NbDoc}{D_k}) + 1$$

Où $NbDoc$ est le nombre total de documents du corpus du système, et D_k est le nombre de documents contenant t_k.

Les termes résultant des trois opérations précédentes sur un document sont appelés **mots-clés.** Ils forment avec leur poids le représentant du document en question. Le même procédé est effectué sur la requête pour obtenir son représentant. Un représentant d'un document est donc une liste de couple (mc, p) ou p et le poids du mot-clé mc.

4. Un **index :** Une fois les représentants des documents construits, le SRI construit un index qui sert à repérer rapidement les documents contenant au moins un des mots-clés qui apparaissent dans la requête.
Un *index* est une structure qui permet d'associer, à chaque terme d'indexation, la liste des documents qui contiennent ce terme. En plus de la simple relation d'appartenance d'un terme à un document, un index peut fournir d'autres informations notamment le poids du terme dans le document, la co-occurrence des termes dans un document, la position du terme dans le document ou sa position dans telle ou telle unité logique (titre,

résumé, titre de chapitre,...), etc. La structure d'index la plus utilisée est celle qu'on appelle *le fichier inversé*. Ce fichier est représenté comme suit :

$mc_1 \longrightarrow$	$d_i \longrightarrow$	$d_j \longrightarrow$	$\cdots$	$\cdots$	$\cdots$	d_k
$mc_2 \longrightarrow$	$d_l \longrightarrow$	$d_m \longrightarrow$	$\cdots$	d_n		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		
$mc_n \longrightarrow$	$d_j \longrightarrow$	$d_k \longrightarrow$	$\cdots$	$\cdots$	d_m	

Pour chaque mot clé mc_i correspond une liste de documents où il apparaît. On appelle **indexation** l'ensemble des opérations que nous venons de décrire à savoir la construction des représentants des documents et de la requête et la construction de l'index.

5. Une **fonction d'appariement (correspondance) :** L'index construit, l'appariement entre la requête et l'index peut désormais s'effectuer. L'appariement consiste à détecter les documents pertinents pour la requête posée, et éventuellement calculer le degré de cette pertinence appelé aussi score. Ce score sert à trier la liste des documents retournés.

   Sous-jacent à la mesure de ce score ou à l'extraction des documents pertinents, on rencontre la notion de **pertinence**. Cette dernière a différents sens selon qu'on la considère du point de vue de l'utilisateur ou du point de vue du système lui-même. Un utilisateur juge un document pertinent s'il répond à son besoin d'information et évidemment, ce jugement est subjectif, variant d'un utilisateur à un autre pour une même requête.
   Un système considère un document pertinent s'il répond aux critères spécifiés dans la requête. Ce jugement ne dépend que des représentations attribuées aux documents et à la requête ainsi que de l'algorithme d'appariement.

   Le but de tout SRI est de minimiser au maximum la perte d'information pendant toutes les étapes du processus de RI pour rapprocher au maximum la pertinence vue par l'utilisateur de celle vue par le système. La figure 1.1 montre le processus général de la RI.

6. Un **Modèle de recherche d'information (MRI) :** Lorsque l'on parle de MRI, on englobe les notions d'index et de fonction d'appariement. La fonction d'appariement utilise des informations contenues dans l'index. Ce dernier doit donc être construit en rapport avec cette fonction. Le MRI prédit les documents qui sont pertinents (et calcule leur degré de pertinence) et les documents qui ne le sont pas. Il existe essentiellement quatre modèle de RI classiques : le modèle booléen, le modèle vectoriel, le modèle probabiliste et les modèles logiques. Ces modèles sont détaillés dans [BY99].

7. Un **moteur de recherche** est une machine spécifique (matérielle et logicielle) capable de construire, sur la base d'un corpus de documents, un

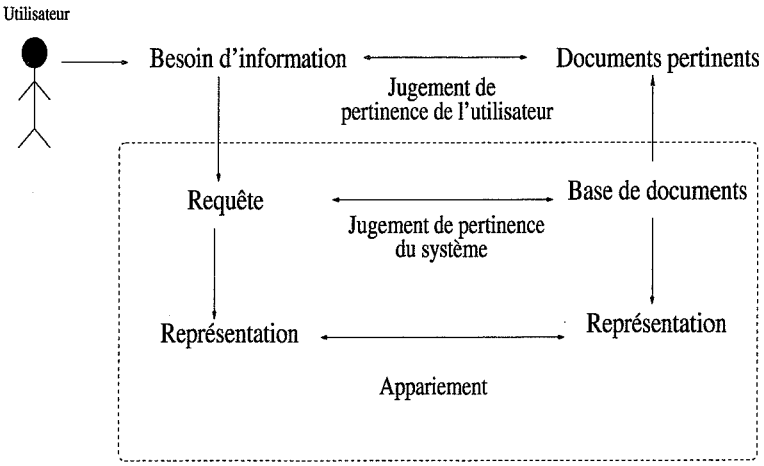


FIG. 1.1 – *Processus de recherche d'information*

index approprié, et ensuite offrir à l'utilisateur le moyen de rechercher dans ce corpus. Après la saisie d'une requête, le moteur applique la fonction de correspondance entre l'index et la requête, et retourne à l'utilisateur une liste de résultats (voir la figure 1.1).

8. La **performance** et l'**efficience :** Nous convenons dans ce rapport de thèse d'appeler performance d'un SRI sa capacité à satisfaire l'utilisateur en terme de pertinence des documents retournés. Nous parlerons de l'efficience d'un SRI lorsqu'il s'agit d'évaluer tous les critères autres que la performance, à savoir, le temps de réponse, le coût et le temps de l'indexation de son corpus, la facilité d'utilisation, la simplicité de l'interface utilisateur, etc.

9. Le **bruit** et le **silence :** Dans la RI deux concepts sont utilisés afin de décrire la performance d'un système. Le bruit, dans une réponse représente tous les documents non pertinents à la requête. Le silence, dans une réponse, représente tous les documents pertinents du corpus du système n'ayant pas été retrouvés pour cette requête.

1.1.2 Système de recherche d'information centralisée (SRIC)

La figure 1.2 illustre les principales composantes d'un système de recherche d'information centralisée (SRIC). Un SRIC fonctionne d'une manière totalement centralisée. Sa particularité est l'unicité de son index. Tous les documents de son corpus sont indexés dans le même fichier ou la même structure qui sert de base pour la recherche.

La particularité d'un SRIC, qui est l'unicité de son index, possède un inconvénient majeur : la non-extensibilité. En effet, vu la quantité phénoménale des

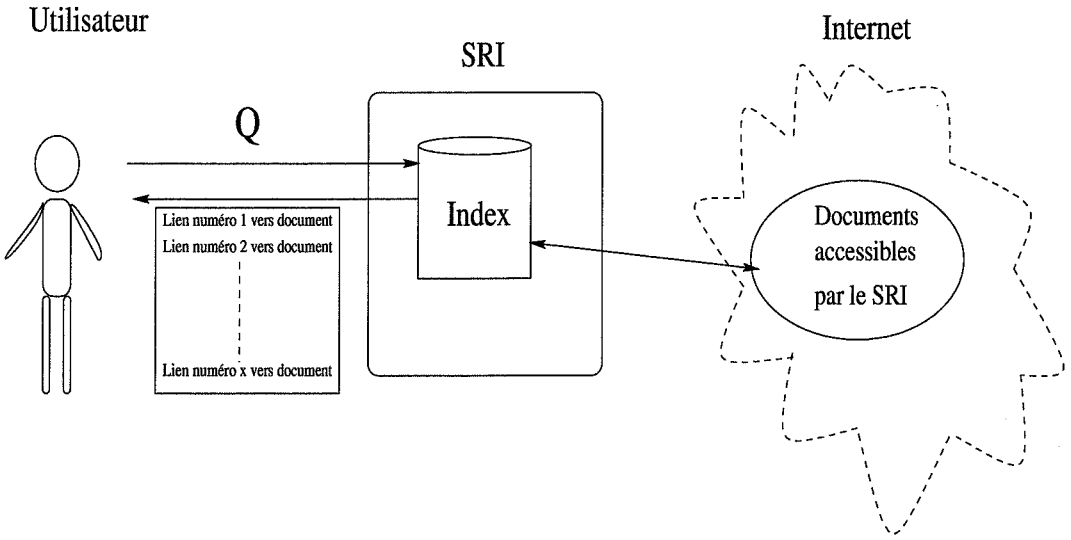


FIG. 1.2 – *Système de recherche d'information centralisée (SRIC)*

données sur Internet, et face au rythme de croissance de ces données, il est impensable d'employer un index unique pour toute cette quantité de données à cause des problèmes suivants :

1. la puissance et les capacités de sauvegarde n'évoluent pas avec la même vitesse que les données sur Internet ;
2. il est impossible de se procurer tous les documents existants. En effet, il existe sur Internet des documents inaccessibles comme par exemple :
   - les documents protégés par mot de passe ;
   - les documents des sites qui interdisent l'indexation de leurs documents par le biais du fichier *robots.txt* ;
   - les documents de la partie cachée d'Internet, correspondant souvent à des documents générés dynamiquement comme, par exemple, les horaires de trains ou les cours boursiers ;
   - ou tout simplement, les documents qui ne sont pas publiés sur le Web tels que les mèls, les fichiers sur les disques dur, etc.
3. il est très coûteux de construire un index pour la totalité des documents collectés, car il est nécessaire de les charger localement afin de les indexer. Or, ce chargement ralentit le fonctionnement du réseau et les serveurs à partir desquels le chargement s'effectue ;
4. la mise à jour de l'index unique n'est pas facile, et cette tâche peut durer longtemps. Or, certains sites, comme, par exemple les sites d'information, mettent leurs documents à jour quotidiennement ;

5. l'unicité de la méthode d'indexation malgré l'hétérogénéité des sources des documents est inadéquate. En effet, des documents rédigés dans des langues différentes sont indexés de la même façon.

1.1.3 Système de recherche d'information distribuée (SRID)

Un système de recherche d'information distribuée (SRID) a le même but qu'un SRIC, à savoir, satisfaire le besoin d'information de l'utilisateur. Couvrir au maximum la masse d'information existante sur Internet est un premier pas vers ce but. En effet, la performance d'un SRI dépend de deux critères : sa couverture de l'information disponible sur Internet et le MRI utilisé. Effectivement, un grand corpus riche en documents pertinents pour une requête donnée n'est pas exploitable si son MRI n'est pas performant pour retrouver ces documents. De plus, un MRI efficace ne peut retrouver des documents pertinents n'existant pas dans le corpus qu'il a indexé.

Les SRIDs ont été conçus pour assurer une grande couverture de l'Internet et pour pallier les problèmes rencontrés par les SRICs. Un SRID est constitué d'un courtier qui communique avec un ensemble de serveurs (voir la figure 1.3). Chaque serveur correspond à un SRI. Chaque serveur contient donc un index et un moteur de recherche qui exploite cet index.

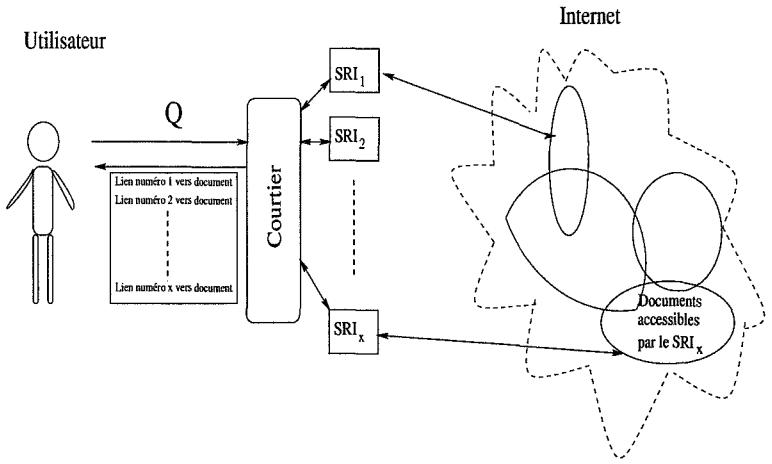


FIG. 1.3 – *Système de recherche d'information distribuée (SRID)*

En clair, un SRID est un ensemble de serveurs qui coopèrent afin d'atteindre un but commun, celui de répondre au mieux aux besoins d'information des utilisateurs. L'idéal est que le fonctionnement interne de cet ensemble de serveurs reste transparent à l'utilisateur qui garde l'impression de travailler avec un seul serveur, c'est à dire le courtier.

Les SRIDs possèdent l'avantage d'éviter à l'utilisateur de consulter indépendamment chaque serveur. Imaginons un utilisateur qui cherche des articles scientifiques sur un domaine particulier. Il est fort probable que les articles recherchés se trouvent dispersés sur plusieurs sites d'universités ou d'éditeurs de revues scientifiques. Dans ce cas, l'utilisateur peut envoyer sa requête à chacun de ces sites (en supposant qu'il les connaisse) et examiner les documents retournés par chacun d'eux. Ceci est une tâche fastidieuse. Premièrement, l'utilisateur devra se familiariser avec les diverses interfaces et langages d'interrogation de chaque site. Deuxièmement, il devra consulter chaque liste retournée (avec probablement des duplicatas à travers les listes). Enfin, l'utilisateur sera submergé par le nombre de documents retournés si le nombre de sites interrogés s'avère important.

Le processus de recherche d'un SRID classique se déroule comme suit. Le courtier joue le rôle d'interface entre l'utilisateur et les serveurs. Le courtier reçoit donc la requête, la transmet aux serveurs qu'il juge les plus susceptibles de contenir des documents pertinents à la requête. Les serveurs interrogés traitent parallèlement la requête et y répondent par des listes de documents. Le courtier enfin fusionne ces différentes listes reçues et présente à l'utilisateur une liste unique de réponses.

1.1.4 Les SRIDs : une solution pour les problèmes des SRICs

Les SRIDs, en distribuant le processus d'indexation et de recherche, apportent des solutions aux problèmes posés par les SRICs :

1. La couverture maximale de l'information disponible sur Internet n'est plus une tâche qui incombe à un seul serveur. La couverture obtenue est l'union des couvertures des serveurs du SRID. Un SRID permet également d'exploiter efficacement les ressources matérielles puisque la tâche de stockage est distribuée sur un ensemble de serveurs.

2. Une source de documents[1] ne souhaitant pas être indexée par d'autres serveurs distants, peut installer son propre SRI et s'insérer dans le SRID. De cette façon, ses documents sont visibles sans qu'ils ne soient pour autant indexés par des serveurs "étrangers". C'est le cas par exemple des serveurs payants, qui souhaitent être référencés sans que le contenu de leurs documents ne soit consultables.

3. Les serveurs adhérant au SRID, sont responsables de la construction et de la mise à jour de leur corpus. La décentralisation de ces deux tâches les rend moins coûteuses.

4. Chaque serveur étant responsable de l'indexation de sa collection, il est libre de choisir la méthode d'indexation appropriée à son corpus (ou collection), par exemple en fonction de la langue.

1. Une source de documents est un ensemble de documents mis à disposition par un particulier ou une organisation désirant les publier.

1.2 Problématique

Nous détaillons dans cette section les problèmes posés par la recherche d'information distribuée (RID). Le premier problème est le repérage des serveurs qui vont adhérer au SRID. Le second problème concerne la décision de reconstruire les collections des serveurs ou de les garder telles quelles. En d'autres termes, il s'agit de choisir entre, construire un SRID à base de serveurs autonomes (possédant leur propres collections et leur propre SRI) déjà existants, ou construire un SRID à base de nouveaux serveurs où l'on place des collections construites à partir des collections des serveurs existants (serveurs ayant bien voulu fournir leur collections au courtier). Le troisième problème est de choisir les données susceptibles d'indiquer l'utilité d'un serveur à une requête donnée afin d'effectuer la tâche de sélection. Le quatrième problème concerne la manière d'utiliser les données dont le courtier dispose afin d'effectuer la sélection de serveurs. Le cinquième problème est, pour le courtier, de déterminer le nombre de documents, provenant d'un serveur interrogé, à inclure dans la liste de réponse à retourner à l'utilisateur. Un autre problème est celui de la communication entre le courtier et les serveurs sous-jacents. Enfin, le dernier problème est celui de la fusion des résultats retournés par les différents serveurs interrogés afin de produire une seule liste de résultats à présenter à l'utilisateur du SRID.

1.2.1 Repérage des serveurs

Habituellement cette opération est effectuée manuellement, c'est-à-dire, par souscription. Un serveur désirant mettre à disposition sa collection s'inscrit au niveau du courtier. Aucun travail à notre connaissance n'a été fait à ce jour pour automatiser cette tâche. Néanmoins cette opération relève du domaine organisationnel plutôt que du domaine technique.

1.2.2 Reconstruire les collections vs. garder les collections d'origine

Les SRID peuvent se classer en deux catégories : les SRIDs avec des collections gérées indépendamment, et les SRIDs avec gestion centralisée des collections. Dans le premier cas, les serveurs gèrent les collections de manière indépendante et restent maîtres de leurs collections. Dans le deuxième cas, les collections sont fournies au courtier qui s'occupe de les fusionner en une seule collection et ensuite de subdiviser cette dernière. La subdivision peut se faire selon deux modes :

- *le partage des documents :* Chaque serveur se voit attribuer un ensemble de documents qui constituera sa collection. Les collections peuvent représenter des regroupements de documents qui satisfont certains critères comme, par exemple, une collection peut contenir des documents d'un

même type (image, texte, sons, etc.) ou des documents qui proviennent d'une même source ou d'une même région géographique etc. Les collections peuvent également être réparties selon divers thèmes (histoire, littérature, économie, etc.). Ce type de regroupement semble augmenter l'efficacité du système [XC99].

- *le partage des termes d'index:* Le système construit un index global qui sera ensuite découpé pour que chaque serveur se voit attribuer une partie de l'index et l'ensemble de documents relié à cette partie. Chaque serveur est alors associé à l'ensemble de termes contenus dans la partie de l'index qui lui a été attribuée. La requête également sera divisée en une ou plusieurs sous-requêtes, correspondant chacune aux termes relatifs à un même serveur. Les sous-requêtes seront alors envoyées aux serveurs appropriés. Dans ce cas de partage c'est le temps de réponse qui est favorisé [RNB98].

Notons néanmoins que dans le deuxième cas, les avantages des SRIDs cités dans la section 1.1.4 ne sont pas tous valables. En effet, le processus de construction des collections étant centralisé, le seul avantage qui reste valable est celui relatif à l'augmentation de la couverture de l'information sur Internet.

1.2.3 Choix des données nécessaires à la sélection des serveurs

Comment choisir les informations qui permettent de prédire la pertinence d'un serveur à une requête donnée? En d'autres termes, comment pouvoir déterminer si un serveur contient des documents pertinents en réponse à la requête de l'usager? C'est principalement sur cette question que nous nous sommes penchés dans ce travail de thèse.

Si l'on ne veut pas faire un choix aléatoire des serveurs qu'on doit interroger pour une requête donnée, il est nécessaire d'avoir certaines informations sur lesquelles nous nous baserons pour effectuer le choix. La nature de ces données dépend des critères sur lesquelles la sélection sera faite. Si, par exemple, la sélection repose sur la proximité géographique, alors les données appropriées contiendront des informations relatives à la position géographique des serveurs.
Mais dans la grande majorité des cas, les méthodes de sélection se basent sur le contenu des collections afin de décider de l'utilité d'un serveur pour une requête donnée. Dans ce but, on peut construire, au préalable, des représentants des serveurs. Chaque représentant est un ensemble de données contenant des informations sur le contenu de la collection associée au serveur. Ces données peuvent être une description textuelle du contenu de la collection ou des données statistiques sur l'ensemble des termes qui apparaissent dans les documents de la collection comme, par exemple, leur fréquences d'apparition, le nombre de collections du SRID contenant chacun de ces termes, etc. À la réception d'une requête, le courtier compare les termes de cette requête aux différents représentants des serveurs. Le jugement sur l'utilité de ces derniers peut alors s'effectuer selon diverses méthodes.

Une remarque préliminaire est qu'une telle approche postule une coopération de la part des serveurs. En effet, on ne peut pas avoir de renseignements sur le contenu d'une collection sans l'aide du serveur qui l'héberge.

Une deuxième remarque s'impose. Une fois construit, le représentant d'un serveur donne des informations sur le contenu de la collection gérée par le serveur, non sur la capacité du serveur à retourner des documents pertinents pour une requête donnée. La difficulté, dans ce cas, est qu'un serveur contenant des documents pertinents mais dont le moteur de recherche est médiocre pourra être sélectionné pour interrogation alors qu'il s'avère peu ou pas capable de retourner les documents pertinents qu'il contient.

Une dernière remarque porte sur les données elles-mêmes. On peut se demander, par exemple, si la fréquence des termes dans une collection est vraiment appropriée pour estimer le nombre de documents pertinents à une requête? Supposons une requête 'société', comment pourrait-t-on définir quelle est la collection la plus pertinente parmi les collections suivantes : celle qui contient 100 occurrences du terme 'société', celle qui en contient 10 ou celle qui en contient 5? De plus, si la première des collections citées (en l'occurrence celle où apparaît 100 fois le terme 'société') contient 4 documents contenant chacun 25 occurrences de 'société', la seconde contient 1 document contenant les 10 occurrences et enfin la dernière collection contient cinq documents contenant une occurrence de 'société'. Si l'on juge selon le nombre de documents pertinents (en supposant qu'un document est pertinent si le terme 'société' y apparaît au moins une fois) alors la dernière collection s'avère la plus pertinente alors que le terme 'société' y apparaît le moins.

Plusieurs questions nous viennent alors à l'esprit :

- quelle est la nature des données qui pourraient nous informer, d'une manière fiable, sur le contenu des collections?
- quelles sont les informations qui vont nous permettre de tester la performance des SRI utilisés par les serveurs?
- y-a-t-il un moyen de se procurer ces informations sans la coopération des serveur?

Autant de questions auxquelles nous tentons d'apporter une réponse dans nos travaux.

1.2.4 Sélection des serveurs

Une fois récupérées, les données qui servent à sélectionner les serveurs doivent être utilisées efficacement afin de prédire, et de manière fiable, la pertinence des serveurs pour la requête en cours. La plupart des méthodes de sélection

existantes adoptent la méthode de classement des serveurs selon leur degré de pertinence. Dans ce cas, un score est associé à chaque serveur. Celui-ci est calculé sur la base des données disponibles au niveau du courtier. Les serveurs étant classés selon leur score, quelques possibilités de sélection peuvent être envisagées. Par exemple, interroger les n_serv premiers serveurs les mieux classés, ou interroger les serveurs dont le score dépasse un certain seuil fixé, noté s. Mais alors, comment choisir l'une ou l'autre de ces constantes n_serv ou s afin de bien dissocier les serveurs pertinents de ceux qui ne le sont pas. Est-il logique d'utiliser les mêmes valeurs quelque soit la requête ou faut-il les calculer en fonction de cette dernière?

1.2.5 Sélection des documents

Une fois l'étape de la sélection de collections franchie, l'ensemble des serveurs sélectionnés sont interrogés. Il reste alors à sélectionner les documents à retenir parmi ceux retournés par chaque serveur. Dans ce cas, l'objectif est de retrouver le plus grand nombre de documents pertinents de chaque serveur tout en minimisant le nombre de documents sans intérêt. En supposant que les serveurs classent tous leurs documents par ordre de pertinence, le choix de ces documents peut s'effectuer selon plusieurs approches :

- Tous les documents rendus par un serveur interrogé sont retournés à l'utilisateur. Dans ce cas, il n'y a pas de risque d'éliminer un document pertinent qui se trouverait dans l'une des listes retournées par les serveurs. Cependant, ces dernières peuvent être longues et par conséquent leur chargement intégral prolonge le temps d'attente de l'utilisateur. En outre, le bruit (la présence de documents sans intérêt à l'usager) risque d'être plus important.

- Déterminer le nombre de documents à retrouver à partir de chaque serveur interrogé. Ce nombre peut être calculé de plusieurs manières.
  La première est la plus simple, l'utilisateur détermine un nombre pour chaque serveur. Mais cette approche n'est pas très recommandée vu la subjectivité du choix de l'utilisateur.
  La deuxième solution consiste à définir un nombre fixe pour tous les serveurs. Là encore, la solution n'est pas très rationnelle car certains serveurs peuvent être meilleurs que d'autres pour une requête donnée.
  Une troisième solution est de mettre en place un processus d'apprentissage, déterminant le nombre pour chaque serveur, par rapport à des expériences passées. Le risque, dans ce cas, est de favoriser les serveurs non pertinents pour la requête en cours mais qui auraient répondu correctement pour les requêtes d'apprentissage.
  Enfin une dernière possibilité consiste à déterminer le nombre pour chaque serveur en fonction du score du serveur, score obtenu dans l'étape de la sélection.

- Dans le cas où les scores des documents sont retournés par les serveurs, on peut définir un seuil pour chaque serveur et sélectionner tous ses documents dont le score dépasse ce seuil. La difficulté, dans ce cas, est de définir la valeur du seuil de manière à ce que la sélection ne soit ni trop restrictive ni trop large.

- Finalement, on peut charger les documents retournés par les serveurs interrogés et recalculer leur score (degré de similarité entre le document et la requête). Cette solution est la plus coûteuse comparées aux précédentes en terme de temps d'exécution. Néanmoins, elle permet de différencier avec beaucoup plus de précision les documents pertinents de ceux qui ne le sont pas.

Alors, une sélection des documents est-elle opportune? Si oui, quelle est la méthode à utiliser? Cette problématique est également étudiée dans cette thèse. Par contre, nous la considérons indissociable du processus de sélection de serveurs. En effet, le but dans les deux cas est de maximiser le nombre de documents pertinents à présenter à l'utilisateur.

1.2.6 Communication entre courtier et serveurs

Chaque serveur est souvent différent par la structure, la dimension et le contenu de sa collection, la méthode d'indexation et de recherche utilisée et, pour ce qui concerne la communication, du langage d'interrogation supporté et de la syntaxe des réponses qu'il retourne. En l'absence d'adoption d'un protocole de communication par le SRID, le courtier doit s'adapter aux spécificités de chacun des serveurs sous-jacents. Est-il plus intéressant d'instaurer un protocole de communication, sachant que cette solution exige la coopération des serveurs?

1.2.7 Fusion des résultats provenant des différents serveurs interrogés

Pour qu'un utilisateur ait la même vision d'un SRID que d'un SRIC, le courtier fusionne les résultats retournés par les serveurs interrogés en une seule liste de réponses. Idéalement, les documents sont triés par ordre décroissant de leur pertinence. Mais cet idéal est difficile à atteindre à cause de l'hétérogénéité des SRIs des serveurs. En supposant que les serveurs retournent des listes de documents triés selon leur score, et que ces scores sont disponibles, les documents des différentes listes ne peuvent être fusionnés par rapport à ce score. En effet, ces derniers ne sont pas comparables à cause des différences des méthodes de calcul de score, des modèles de pondération, des méthodes d'indexation, etc. entre les serveurs. Il est donc nécessaire de trouver un autre moyen de fusionner ces listes. Dans cette thèse, nous avons adopté une méthode de fusion élaborée par Y. Rasolofo [RAS01] qui utilise la longueur des listes de réponses ainsi que le

rang de chaque document comme critère pour définir la position d'un document dans la liste finale retournée à l'utilisateur.

1.3 Objectifs

Nous avons cité dans la section précédente les différents problèmes auxquels on est confronté lorsque l'on veut concevoir un SRID. Dans cette thèse, nous avons choisi de nous concentrer essentiellement sur le problème de sélection de serveurs.

Nos travaux s'inscrivent alors dans la perspective de développer une méthode de sélection de serveurs qui aura les caractéristiques suivantes :

1. d'un point de vue de la satisfaction de l'utilisateur, le système composé de notre approche de sélection combinée à une méthode de fusion doit atteindre ou dépasser la performance de l'approche centralisée.
2. d'un point de vue pratique, notre approche doit être souple et la moins contraignante possible. Nous définissons une telle approche comme étant celle :
   - qui n'exige pas de coopération entre les serveurs et le courtier ;
   - dont la maintenance est la moins coûteuse possible ;
   - qui soit extensible.

1.4 Contribution

Nous avons développé dans cette thèse une approche d'acquisition des données nécessaires à prédire la pertinence (l'utilité) d'un serveur par rapport à une requête donnée. Cette approche consiste à acquérir l'information nécessaire au moment de l'interrogation plutôt que de sauvegarder au préalable des données sur le contenu des collections. Nous avons, en outre, proposé trois approches de sélection de serveurs. Ces approches fonctionnent sur la base des données acquises précédemment. Nous avons ensuite conduit une étude expérimentale afin de vérifier la validité de nos propositions et de comparer leurs résultats à ceux des approches existantes.

La principale particularité de nos approches réside dans le fait qu'elles tiennent compte de la performance des moteurs de recherche utilisés par les serveurs du SRID. En effet, nous sommes partis du principe qu'un serveur est utile pour une requête donnée s'il est capable de **retourner** des documents pertinents à cette requête. Les méthodes existantes quant à elles considèrent qu'un serveur est pertinent s'il **contient** potentiellement des documents pertinents. En partant de ce principe, nous avons choisi d'analyser les premiers documents retournés par un

serveur afin de lui attribuer un score déterminant son degré d'utilité à la requête. Les serveurs sont alors classés selon ce score. Nos approches de sélection vont ensuite déterminer, à partir du classement des serveurs ceux qui fourniront des documents à retourner à l'utilisateur. De cette manière, nous atteignons notre objectif de concevoir une méthode souple et la moins contraignante possible. En effet, la coopération des serveurs n'est pas nécessaire puisque les représentants des serveurs ne sont pas fournis par les serveurs mais sont extraits des documents retournés par les serveurs. Aucune maintenance n'est exigée par nos approches puisque les représentants sont construits au moment de l'interrogation, ce qui rend également nos approches extensibles. En effet, l'ajout d'un serveur ou sa suppression ne demandent aucune modification si ce n'est son ajout ou sa suppression dans la liste connue par le courtier.
En ce qui concerne notre objectif d'atteindre la performance de l'approche centralisée, les évaluations effectuées dans le chapitre 4 sont encourageantes surtout pour deux de nos trois approches de sélection (nous avons jugé la troisième approche inefficace) lorsqu'elles sont combinées avec une méthode de fusion performante.

1.5 Plan de la thèse

La suite de ce rapport de thèse est structurée comme suit :

- Le deuxième chapitre est composé de deux parties. Une première décrit les étapes d'un processus de recherche d'information distribuée. Une seconde présente une synthèse des différentes approches existantes dans la littérature concernant les étapes de sélection de serveurs, la fusion des résultats ainsi que les approches d'évaluation des SRIDs. Plus de détails sont donnés en ce qui concerne les approches de sélection de serveurs. Un catalogue de ces approches sera dressé suivi d'une étude comparative.

- Le troisième chapitre est consacré au développement théorique de nos trois approches de sélection de serveurs. Nous commencerons par présenter les objectifs que nous nous sommes fixés et les hypothèses que nous avons posées. Nous introduirons par la suite notre approche d'acquisition des représentants des serveurs, suivie des détails de nos approches de sélection de serveurs. Nous finirons par expliquer les motivations de nos choix et les caractéristiques les distinguant des approches suggérées par d'autres auteurs.

- Le quatrième chapitre est dédié à l'étude expérimentale que nous avons conduite afin de vérifier la performance de nos approches et de comparer leurs résultats aux résultats d'autres approches comme par exemple l'approche centralisée.
  Ce chapitre débutera par une présentation de nos collections de test. Nous

détaillerons ensuite la construction des environnements de test, les méthodes expérimentales choisies, les approches de fusion utilisées et les méthodes dont nous nous sommes servis pour évaluer nos approches.
Les expérimentations que nous avons menées sont alors détaillées, nos approches évaluées et comparées aux autres approches.

- Une conclusion suivie des perspectives envisagées terminera ce mémoire de thèse.

Chapitre 2

Recherche d'information distribuée (RID)

Table des matières

2.1 Introduction

Le problème de la recherche d'information est soulevé lorsqu'une personne a accès à une importante masse de documents et se retrouve dans le besoin d'être

aidée pour retrouver l'information pertinente, d'où l'apparition des systèmes de recherche d'information (SRI). La forme la plus simple d'un SRI est celle qui permet à l'utilisateur de formuler sa requête, traite cette dernière et propose à l'utilisateur une liste de documents (ou de références à ces documents) jugés pertinents. D'autres formes existent comme par exemple les systèmes d'aide à la navigation ou les annuaires.

Le problème de la RID est soulevé lorsque l'utilisateur a accès à plusieurs SRIs, chacun œuvrant souvent sur une collection différente, et a besoin d'être aidé pour retrouver l'information pertinente, sans avoir à interroger séparément chaque SRI. L'utilisateur d'un système de recherche d'information distribuée (SRID) envoie sa requête à un seul système qui se charge d'interroger les différents SRIs.

Ce deuxième chapitre a pour but d'introduire le domaine de la recherche d'information distribuée dans lequel se situe cette thèse. Dans la section suivante nous présentons d'abord la terminologie qui sera adoptée dans l'ensemble de ce travail, nous présentons ensuite les différents composants fonctionnels et physiques d'un SRID ainsi que les problèmes subordonnés à la RID. Nous détaillons ensuite dans les sections 2.3, 2.4 et 2.5 les points qui ont rapport à notre travail, à savoir l'évaluation des SRID, la sélection des serveurs et la fusion des résultats. En outre, et vu que cette thèse traite de la sélection de serveurs, nous présentons une étude comparative dans la section 2.4.2 des différentes méthodes de sélection de serveurs publiées.

2.2 Les systèmes de recherche d'information distribuée (SRID)

2.2.1 Terminologie

Définissons d'abord les termes qui vont nous servir à présenter le domaine de la RID.

1. Une **source d'informations** est un ensemble de documents mis à disposition par un particulier ou une organisation désirant les publier.
2. Une **collection** est un ensemble de documents appartenant à une ou plusieurs sources d'information.
3. Une **liste de résultats** est une liste triée de références vers des documents qui sont jugés pertinents par le système à une requête donnée. Ces listes peuvent avoir des syntaxes très différentes et peuvent contenir des informations variées telles :
   - les degrés de pertinence des documents ; le degré de pertinence peut être une valeur numérique ou une valeur symbolique telles que des graphiques ou des étoiles etc.

- les passages qui contiennent le plus de termes communs avec la requête ;
- les résumés des documents ;
- etc.

4. Un **serveur** est une machine qui offre à un utilisateur le service de rechercher de l'information dans une collection qu'il héberge. Un serveur contient un SRI local qui effectue cette tâche. L'utilisateur peut interroger directement le serveur ou depuis une autre machine en passant par un réseau.

5. Un **courtier** est un module qui joue le rôle d'intermédiaire entre l'utilisateur et les différentes sources d'information. Le courtier agit comme un pseudo-moteur de recherche qui reçoit en entrée une requête et fournit en sortie une liste de résultats. Le fonctionnement du courtier sera présenté en détails dans la section suivante.

6. Un **représentant de serveur** est un ensemble d'informations détenues par le courtier et décrivant certaines caractéristiques du serveur comme par exemple les termes apparaissant dans la collection du serveur, leurs fréquences d'apparition, ainsi que des informations utiles à son interrogation (sa localisation, son coût, ses droits d'accès, etc.).

7. L'**utilité d'un serveur** à une requête se traduit, selon nous, par sa capacité à *contenir et retourner* des documents pertinents à cette requête.

8. La **fréquence document** d'un terme t_j est le nombre de documents, dans un collection, contenant t_j.

9. La **fréquence collection** d'un terme t_j est le nombre de collections, dans un SRID, contenant t_j.

2.2.2 Fonctionnement d'un SRID

Un SRID simple est composé de plusieurs serveurs et d'un courtier (figure 2.1). Le courtier est le cœur d'un SRID. Le courtier contient cinq composants logiciels : un module de gestion, un module frontal (une interface utilisateur), un module de sélection de serveurs, un module de communication et un module de fusion de résultats.

D'une manière globale, le *module de gestion* construit les représentants des serveurs du système dont il connait les spécificités. L'utilisateur soumet sa requête par le biais du *module frontal (interface utilisateur)*, le *module de sélection de serveurs* choisit les serveurs qui sont susceptibles de répondre à cette requête par des documents pertinents, le *module de communication* adapte alors la requête au langage d'interrogation de chaque serveur sélectionné, leur achemine la requête et réceptionne leur réponse. Enfin, le *module de fusion des résultats*

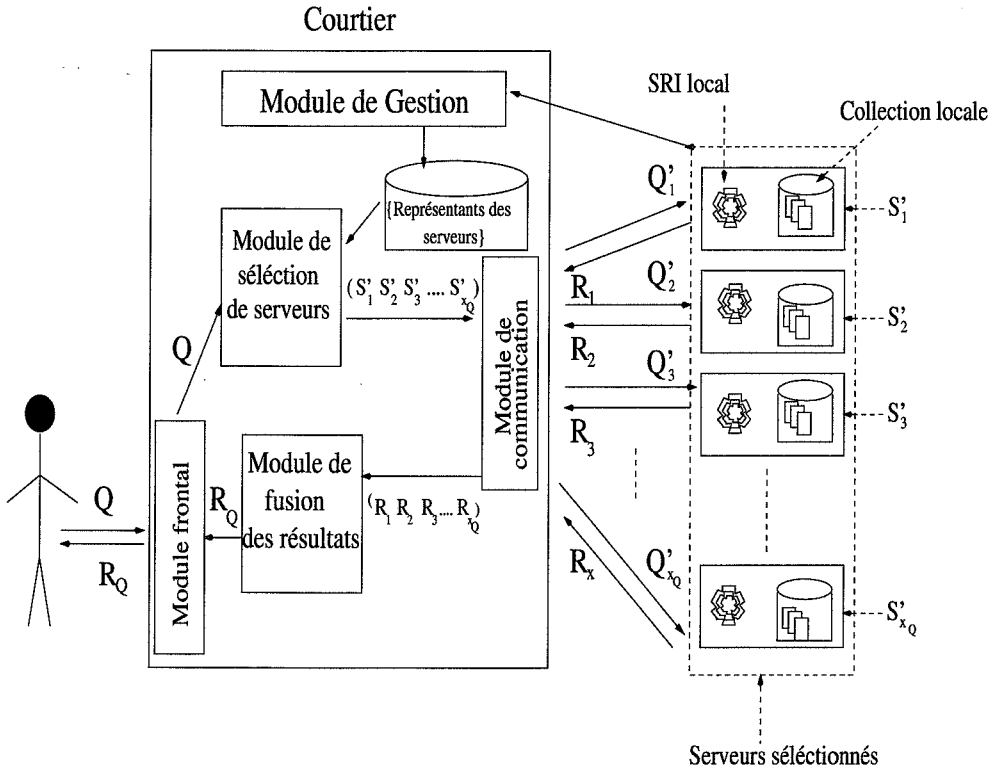


FIG. 2.1 – *Les modules fonctionnels d'un système de recherche d'information distribuée (SRID)*

groupe les réponses reçues par le module de communication et rend via l'interface utilisateur une liste unique de résultats à l'utilisateur. Nous détaillons dans ce qui suit le fonctionnement de chaque module. Pour cela nous adoptons quelques notations, soient :

- Q une requête soumise par un utilisateur ;
- S l'ensemble des n serveurs connus par le système ;
- *Rep* l'ensemble des représentants des serveurs connus par le système ;
- S'_Q l'ensemble $\{S'_1, S'_2, \ldots, S'_{x_Q}\}$, le sous ensemble de S des serveurs sélectionnés pour Q, et x_Q son cardinal ;
- Q'_i est la requête traduite dans le langage d'interrogation du serveur S'_i ;
- R_i la réponse de S'_i, R_i est la liste de couples $< D, info >$ provenant de S'_i, où D est une référence vers un document de la collection de S'_i jugé pertinent par S'_i pour Q et *info* des informations relatives à ce document ;
- R_Q la liste finale de résultats retournée à l'utilisateur.

2.2.2.1 Le module de gestion

Ce module s'occupe essentiellement de construire *Rep* (voir la figure 2.1) l'ensemble des représentants des serveurs et le maintenir à jour. L'automatisation de cette tâche est très importante vue la rapidité avec laquelle les documents paraissent, sont modifiés et disparaissent. En outre, les approches qui se basent sur une construction manuelle [CH95] ou une construction des représentants par apprentissage [Voo95] sont difficiles et coûteuses à appliquer dans un environnement en continuel changement.

2.2.2.2 Le module frontal

Ce module se charge d'établir la communication entre l'utilisateur et le système. Ses fonctionnalités varient d'un système à un autre, il peut offrir une aide à l'utilisateur pour formuler son besoin d'information, pour choisir les sources d'information à rechercher, pour définir le nombre de documents qu'il voudrait retrouver, pour affiner sa requête de manière manuelle ou automatique, etc.

2.2.2.3 Le module de sélection de serveurs

À partir de *Rep*, le module de sélection détermine S'_Q (voir la figure 2.1), l'ensemble des serveurs qui sont les plus susceptibles de contenir des documents pertinents à la requête.

Certes la sélection manuelle est possible. Dans un tel cas, les utilisateurs débutants ont tendance à sélectionner tous les serveurs. Mais le propos de cette thèse est justement d'implémenter et de vérifier l'efficacité d'une sélection automatique.

Certains courtiers ne procèdent pas à une sélection et interrogent tous les serveurs disponibles. Cette approche est la plus répandue dans les méta-moteurs de recherche d'information sur Internet.

La plupart des méthodes de sélection existantes procèdent au classement des serveurs selon leur utilité à la requête qui est calculée par un score. Le score d'un serveur peut être par exemple le nombre présumé de documents pertinents que contient sa collection. Après le classement, la sélection proprement dite, peut s'effectuer selon plusieurs scénarios :

- la sélection fixe : consiste à sélectionner un nombre prédéfini n_serv de serveurs ;
- la sélection par seuil : consiste à sélectionner tous les serveurs dont le score dépasse un certain seuil *seuil* ;
- la sélection économique : consiste à choisir les collections qui satisfont certaines métriques de coût de façon à ce que l'on obtienne le maximum de documents pertinents au moindre coût. Fuhr [Fuh99] propose des métriques basées par exemple sur le coût de traitement d'une requête par une collection et le temps de réponse de cette collection.

Concernant la sélection fixe et la sélection par seuil, la définition de n_serv et de *seuil* soulèvent des difficultés. Car, si l'une de ces valeurs est trop petite des serveurs contenant des documents pertinents peuvent être écartés, ce qui favorise l'augmentation du silence dans la réponse ; et si elle est trop grande, des serveurs ne contenant aucun document pertinent sont aussi interrogés, augmentant ainsi le bruit dans la réponse.

La sélection a plusieurs buts qui peuvent être regroupés en deux catégories. Premièrement, le but d'*efficience*, en d'autres termes, réduire le coût de la recherche en diminuant le nombre de serveurs à interroger. Il est important aussi de réduire la quantité d'information nécessaire au processus de sélection, pour que le système soit extensible. Deuxièmement, le but de *performance* en interrogeant le plus petit nombre de serveurs, sans détériorer la performance du système voire en augmentant sa performance si l'on écarte uniquement les serveurs ne possédant aucun document pertinent.

Un autre type de sélection existe, dans lequel il faut faire le choix des serveurs dupliqués. Dans ce cas la sélection va se faire de plusieurs manières :

- d'une façon aléatoire [KBM94] ;

- selon la topologie du réseau, en choisissant le serveur le plus proche [GS95] ;
- en choisissant le serveur le moins encombré dynamiquement [CC96] ;
- en se basant sur des critères concernant les serveurs, notamment leur temps de réponse, la vitesse de transfert avec ces serveurs, etc. [BAZ+97].

2.2.2.4 Le module de communication

Une fois l'ensemble S'_Q des serveurs à interroger établi, et avant de procéder à l'émission de la requête vers chaque serveur S'_i, le module de communication, en se basant sur des informations sur le langage d'interrogation des serveurs à contacter, convertit la requête dans un format compréhensible par chaque serveur. Pour résoudre le problème de traduction de requête, des protocoles de standardisation ont été suggérés comme, par exemple, *Z39.50* [Org93], *Common Command Language (CCL)* [Neg79] et *ISO 8777* [ISO93].

La deuxième fonction du module de communication consiste à extraire, à partir des R_i, les informations nécessaires à la fusion de ces réponses.

Sur le Web pour le moins, la syntaxe du langage d'interrogation ou la présentation des résultats varient assez fréquemment, nécessitant une mise à jour du module de communication.

2.2.2.5 Le module de fusion de résultats

Le module de fusion regroupe les réponses R_i provenant des serveurs S'_i pour constituer une seule et unique liste de résultats R_Q qui sera présentée à l'utilisateur en réponse à sa requête Q (voir la figure 2.1).

Idéalement, les éléments de R_Q doivent être triés par ordre décroissant de pertinence.

Cependant, et à moins de charger tous les documents référencés par les R_i et de leur attribuer un score, il est difficile de les trier. En effet, les seules informations disponibles dans les R_i sont pour chaque document son rang et éventuellement son score.
Si l'on se trouve dans le cas où le score de chaque document retourné est disponible, nous sommes confrontés à l'incomparabilité des scores. En réalité, chaque serveur a sa propre stratégie de calcul du score, ses propres données, son propre modèle de pondération, et bien d'autres caractéristiques qui lui sont propres, qui de ce fait rendent incomparables les scores de ses documents avec les scores des documents d'autres serveurs.

2.2.3 Architectures possibles d'un SRID

2.2.3.1 Architecture à deux niveaux

L'architecture à deux niveaux est l'architecture dominante et la plus simple (voir figure 2.2(a)), et c'est celle dont il était question dans les sections précédentes. Elle se compose de deux niveaux. Le premier niveau est formé des serveurs connus par le courtier, et qui sont complètement indépendants les uns des autres. Le second niveau est constitué du courtier. Le courtier peut éventuellement être dupliqué afin d'éviter la surcharge de ce maillon.

2.2.3.2 Architecture à plusieurs niveaux

Dans ce cas (voir figure 2.2(b)), plusieurs groupes de serveurs connexes sont formés. Le groupement des serveurs n'est pas physique. La connexité des serveurs regroupés peut porter sur un ou plusieurs critères, comme, par exemple, le thème des collections, leur situation géographique, le type de leurs documents ou leurs sources. Dans ce cas, un courtier est associé à chaque critère. Il contient les méta-données concernant les collections qu'il référence. Un serveur peut être pointé par plusieurs courtiers.
Les courtiers peuvent à leur tour être regroupés et l'opération peut se réitérer sur les groupes de courtiers et former ainsi un système hiérarchique à plusieurs niveaux.
L'avantage de cette architecture est qu'elle est indépendante du nombre de serveurs. En effet l'intégration d'un nouveau serveur ne change par la hiérarchie des courtiers.
On retrouve cette forme d'architecture dans Pharos [DAEA96]. Les courtiers peuvent à leur tour être regroupés et l'opération peut se réitérer sur les groupes de courtiers et former ainsi un système hiérarchique à plusieurs niveaux, comme par exemple dans hGloss [GGMA99].

2.2.3.3 Architecture en graphe

Chaque serveur du système forme un nœud du graphe (voir figure 2.2(c)). Deux nœuds (serveurs) sont liés par un arc si l'un détient des informations concernant l'autre. *À* la réception d'une requête, chaque serveur utilise les informations qui sont en sa possession pour déterminer les serveurs pertinents et leur achemine la requête.
Les informations qu'un serveur détient sur le contenu d'un autre serveur est en général l'index de la collection de ce dernier. L'index d'une collection comme dans le cas, par exemple, du système WHERE [RMF97] qui adopte cette architecture, contient les termes qui apparaissent dans la collection ainsi que trois valeurs max, min et nb. Les deux valeurs max et min représentent respectivement le poids maximum et le poids minimum d'un terme dans tous les documents de

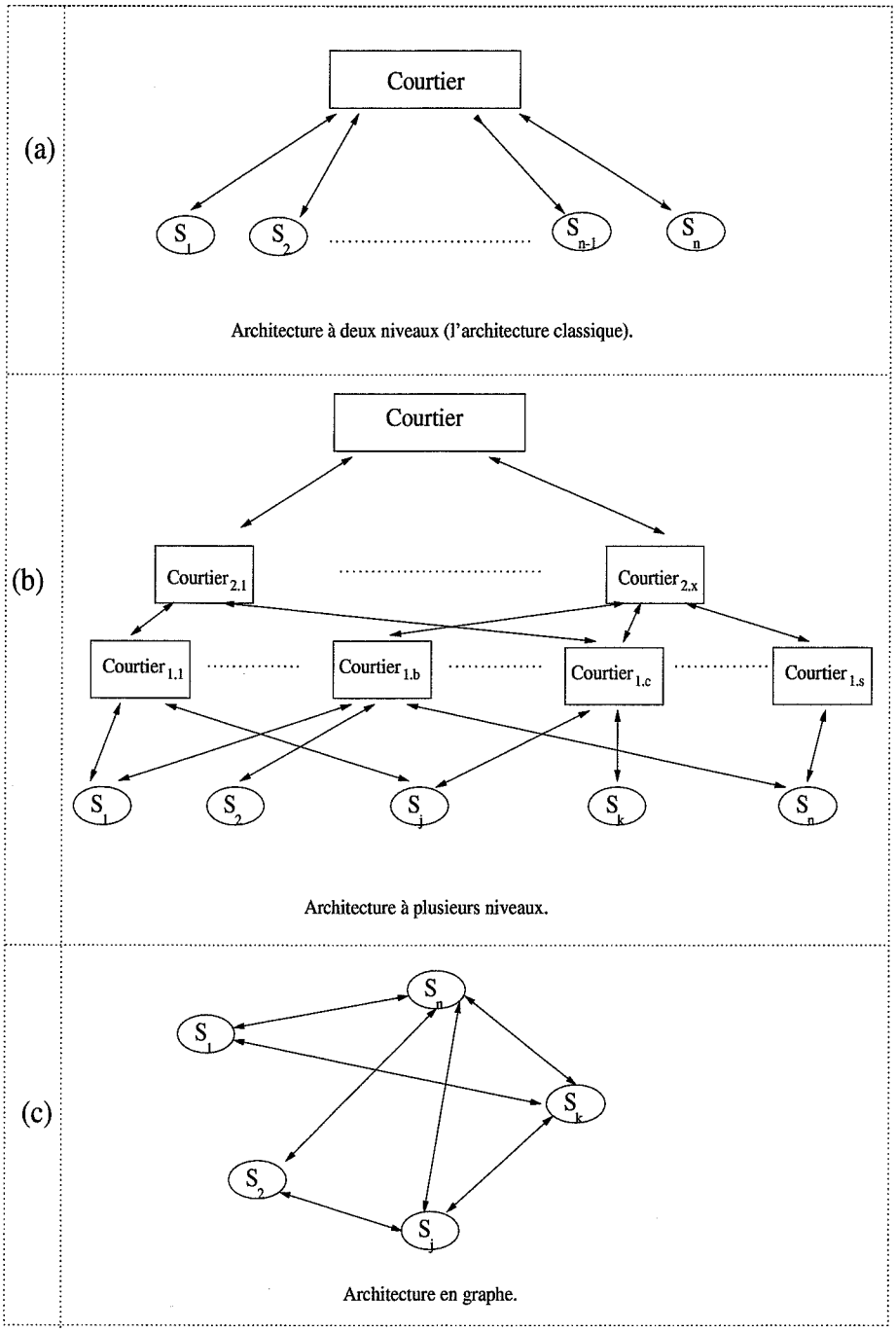


FIG. 2.2 – *Architectures possibles d'un SRID.*

la collection, et *nb* est le nombre de documents de la collection qui contiennent le terme en question.
Cette architecture est adoptée également par les systèmes WHOIS++ [Fou] et ISAAC [RL98].

Cette architecture nécessite la coopération des serveurs qui jouent chacun le rôle de courtier. D'autres inconvénients peuvent être cités, notamment l'espace requis pour sauvegarder les informations nécessaires au routage de la requête vers d'autres serveurs, le temps de traitement de la requête, le trafic à travers le réseau (pour la requête et les résultats), et enfin le problème des boucles infinies ou de la limite du routage, c'est-à-dire, selon quelles conditions la recherche et le routage doivent se terminer. Pour cela, certaines méthodes, comme par exemple, WHERE [RMF97] limitent le nombre de documents à retrouver.

2.2.4 Hétérogénéité des serveurs

Les serveurs sont construits et gérés de façon indépendante, utilisent leurs propres stratégies d'indexation et fonction de correspondance, décident eux-même quand et comment leurs index sont mis-à-jour, évaluent la requête et délivrent la réponse à leur propre rythme, en un mot les serveurs sont *autonomes*.

L'autonomie des serveurs est la cause de plusieurs hétérogénéités, notamment :

1. la méthode d'indexation

   Un serveur peut choisir d'indexer le document en totalité ou en partie, de lemmatiser ou non les termes du document, de supprimer ou non des mots vides (tels que les adverbes, liaisons, etc.). Lorsqu'il s'agit de documents HTML les méthodes d'indexation peuvent porter sur les zones délimitées par des balises particulières telles que, par exemple, *keywords*, *anchor*, etc.

2. la pondération des termes

   L'importance d'un terme dans la représentation ou l'identification d'un document est représentée par une valeur numérique appelée le poids. Il existe une multitude de fonctions de pondération qu'un serveur peut adopter, entre autres :

   (a) la pondération binaire qui indique la présence (poids égal à 1) ou l'absence (poids égal à 0) d'un terme dans un document ;

   (b) la pondération par la fréquence. Le poids d'un terme dans un document dépend du nombre de fois où ce terme apparaît dans ce document ;

   (c) la pondération *tf*idf*: pour calculer le poids d'un terme, on utilise sa fréquence dans chaque document et le nombre de documents dans la collection qui le contiennent.

3. la fonction de correspondance

   Plusieurs fonctions de correspondance peuvent être utilisées pour calculer la similarité entre un document et une requête donnés [Sal89]. Par exemple, certains serveurs peuvent utiliser le calcul du cosinus des vecteurs pour calculer le degré de leur similarité, et d'autres le calcul du produit scalaire des vecteurs.

4. les collections

   Les collections des différents serveurs sont, sans aucun doute, dissemblables ne serait-ce que par leur contenu (collections spécialisées ou généralistes), leur langue et leur taille. Cependant, il peut arriver que certaines collections se chevauchent, comme c'est le cas par exemple des moteurs de recherche commerciaux disponibles sur le Web, qui visent à offrir une couverture complète du Web.

2.2.5 Les principales difficultés dans la RID

Il existe un certain nombre de problèmes concernant les SRIDs. Nous en citons quelques uns. Le premier est lié à la stratégie commerciale des serveurs et les autres sont d'ordre technique.

1. La coopération passive des serveurs

   Par des serveurs qui coopèrent d'une façon passive nous entendons des serveurs qui n'émettent pas d'objection à ce qu'un courtier les interroge et récupère leurs résultats.
   Les administrateurs des serveurs peuvent ne pas approuver le fait qu'un courtier, considéré comme un concurrent, contourne leurs propres résultats. C'est le cas des serveurs Web (moteurs de recherche) qui retournent des résultats incluant des bordereaux publicitaires dont la consultation est plus bénéfique pour eux que la consultation des documents retournés.

2. La coopération active des serveurs

   La coopération active des serveurs signifie que ces derniers acceptent de fournir au courtier des informations qui sont nécessaires pour son fonctionnement. Ces informations peuvent être, par exemple, la technique d'indexation, la fonction de similarité, l'ensemble des termes qui apparaissent dans la collection, leurs fréquences d'apparition, etc.

   La plupart des méthodes proposées dans la littérature supposent la disponibilité, au niveau du courtier, de certaines informations sur le contenu des collections. Or, pour obtenir ces informations, la coopération active des serveurs est très souvent nécessaire, et celle-ci n'est pas systématique pour des raisons de confidentialité ou pour des raisons commerciales.

   Cependant, Callan et al. [CCD99] proposent d'acquérir certaines informations sur le contenu des collections sans recours à la coopération. Cette ac-

quisition consiste à échantillonner le contenu des collections par le processus habituel de soumission des requêtes et de récupération des documents rendus par les collections. Les requêtes d'échantillonnage contiennent un seul terme, ce terme est une entrée du vocabulaire dont le courtier dispose. Cette méthode est coûteuse et limitée quant au type d'informations voulues. En effet, la taille d'une collection par exemple ne peut être obtenue par ce procédé.

3. L'inter-opérabilité

   Le courtier traite avec différents serveurs qui sont autonomes et hétérogènes. L'une des implications qui en découle est la nécessité pour le courtier de pouvoir communiquer et fonctionner conjointement avec les serveurs. Or, ces derniers peuvent changer (langage, présentation, ajout/suppression d'information).

4. La volatilité des documents

   Un document peut voir son contenu évoluer voire être supprimé. Les index locaux des serveurs sont basés sur le contenu des documents à un moment donné. Si certains documents, entre temps, ont subi des modifications ou ont été supprimés, les moteurs locaux pourront retourner des documents qui ne s'accordent pas du tout à une requête ou qui n'existent plus. En outre, et à supposer que le moteur local ait mis à jour son index, le courtier a, pour sa part également, des informations concernant les collections à mettre à jour. Et dans ce cas, seule une coopération parfaite peut résoudre ce problème en informant le courtier des changements qui se sont produits.

   L'un des problèmes relatifs à la volatilité des documents est illustré par le cas suivant : un document indexé par plusieurs serveurs subit une modification de son contenu ; certains des serveurs ont mis à jour leurs index depuis la modification, d'autres non. Le courtier reçoit des scores différents pour le document modifié, même si les mêmes techniques d'indexation, de calcul de scores, etc. ont été utilisées par tous les serveurs. Le courtier dans ce cas n'est pas en mesure de déterminer le score de la version la plus récente du document.

2.3 Méthodes d'évaluation des SRID

En général, un SRI fournit une liste triée selon l'estimation du degré de pertinence des documents par rapport à la requête.
Le but de tout système est de retourner des résultats qui satisferont l'utilisateur. Dans le but d'évaluer différents systèmes ou différentes composantes d'un système, des collections tests ont été créées. De telles collections contiennent un ensemble de documents, un ensemble de requêtes et une liste de jugements de pertinence associée à chaque requête. Le volume de la collection doit être assez important pour que les tests correspondent à la réalité.

Les collections les plus répandues sont celles de TREC (Text REtrieval Conference), de CLEF (Cross-Language Evaluation Forum) et de NTCIR (National institute of informatics Test Collection for Information Retrieval).

Il n'existe actuellement pas de collections de test spécifiques pour l'évaluation des SRIDs. Pour simuler un SRID une approche répandue est de subdiviser l'ensemble des documents d'une collection test en plusieurs sous-ensembles.

L'évaluation d'un SRI en général, ou d'un SRID en particulier, peut porter sur le coût du processus de recherche, le temps de réponse, l'extensibilité du système, la couverture de l'information, le coût des mises à jour, la quantité d'information à transférer etc.
Cependant, dans ce qui suit nous détaillerons uniquement trois types d'évaluation de SRID : l'évaluation de la performance générale du système (i.e. évaluation du résultat final), l'évaluation des méthodes de sélection et l'évaluation des méthodes de fusion. Cette dernière est assimilée à l'évaluation de la performance générale du système.

2.3.1 Évaluation classique de performance (*EvalPerf*)

EvalPerf consiste à évaluer un SRID comme un SRIC : le processus de recherche incluant la sélection, la communication et la fusion des résultats est évalué comme un tout. Cette évaluation utilise les métriques de la RI classique, à savoir la *précision* et le *rappel*. Rappelons que, pour une requête donnée :

- la *précision*, noté P, reflète la capacité du système à ne retourner *que* les documents pertinents, soit :

$$P = \frac{|Pert \cap Ret|}{|Ret|} \text{ avec } | \ Ret \ | \neq 0$$

- le *rappel*, noté R , reflète la capacité du système à retourner *tous* les documents pertinents :

$$R = \frac{|Pert \cap Ret|}{|Pert|} \text{ avec } | \ Pert \ | \neq 0$$

où :

	Pert	est l'ensemble des documents pertinents contenus dans le corpus du système pour une requête donnée ;
	Ret	est l'ensemble des documents que le système a retournés ;
	|E|	est le cardinal de l'ensemble E.

En général, lorsque l'on effectue des tests, plusieurs requêtes sont disponibles et, dans ce cas, c'est la moyenne des valeurs de précision et de rappel pour l'ensemble des requêtes qui est calculée.
Les deux mesures précédentes apportent chacune une information nécessaire à

la détermination de la qualité de recherche qu'un système offre. Mais il n'est pas judicieux de les considérer séparément dans la comparaison de la performance de différents systèmes. En effet, on ne peut faire de jugement de performance lorsque l'on se trouve, par exemple, dans le cas de deux systèmes dont l'un présente une précision meilleure mais un rappel inférieur à celui de l'autre. En outre, le classement des documents pertinents retournés par un système n'est pas pris en compte dans les deux mesures précédentes.

Afin de manipuler conjointement les deux mesures de précision et de rappel, et de prendre en compte le rang des documents pertinents retournés par un système, une troisième mesure est proposée, c'est *la précision moyenne à onze points fixes de rappel* ou *précision moyenne (PM)* [Ris79]. Les onze points de rappel sont fixés à (0, 0.1, ..., 0.9, 1), ce qui signifie que PM est la moyenne des précisions obtenus lorsque l'on présente à l'utilisateur 0%, 10%, ..., 90% et 100% des documents pertinents.
En règle d'usage, une différence de précision moyenne entre deux systèmes n'est significative que si elle dépasse 5%, pour une différence de moins de 5% les deux systèmes sont considérés comme ayant la même performance.

2.3.2 Évaluation de la sélection de collection (*EvalSel*)

Théoriquement, l'évaluation de la sélection de collection revient à déterminer la capacité du système à sélectionner tous les serveurs pertinents et rien que des serveurs pertinents. Cependant, la notion de pertinence (ou d'utilité) d'un serveur n'est pas bien définie. En effet, un serveur pertinent est-il tout serveur dont la collection contient au moins un document pertinent? Ou un serveur qui est susceptible de retourner au moins un document pertinent (si la collection d'un serveur contient un document pertinent mais que le serveur ne peut pas le retourner, il ne sert à rien de considérer cette collection)? Ou un serveur qui, relativement aux autres serveurs, rend beaucoup de documents pertinents? etc. Dans la littérature, toutes les méthodes d'évaluation proposées se basent sur le classement des serveurs. Le classement automatique obtenu par le courtier est comparé au classement idéal. Le classement idéal d'un serveur est le classement par le score idéal des serveurs. Ce dernier dépend de la méthode utilisée pour calculer le score de ce dernier. Si, par exemple, le score d'un serveur représente le nombre présumé de documents qu'il contient, son score idéal est le nombre exact de documents pertinents qu'il contient (ce nombre est disponible grâce aux jugements de pertinence qui font partie des collections de test). Nous explicitons dans ce qui suit quelques unes de ces méthodes.
Remarque : Dans ce qui suit nous considérons S'' comme étant la liste des serveurs (tous les serveurs du système) triée par la méthode de classement à évaluer.

1. Callan et al. [CLC95] proposent de calculer un taux d'erreur du classement des serveurs le **Mean-squared error** (MSE).

$$MSE = \frac{1}{|S|} \cdot \sum_{i \in S} (Id_i - O_i)^2$$

où : Id_i est le rang de S_i dans le classement idéal ;
O_i est le rang de S_i attribué par la méthode de sélection à évaluer.

L'inconvénient de cette métrique est qu'elle ne rend pas compte de l'importance de l'erreur. En effet, le même MSE peut être calculé pour, d'une part, une inversion de rang entre deux serveurs dont la différence entre les nombres de documents pertinents qu'ils contiennent est très grande, et d'autre part, une inversion de deux serveurs dont la différence entre les nombres de documents pertinents qu'ils contiennent est minime.

2. Gravano et al. [GGMA99] proposent d'utiliser pour évaluer le classement des serveurs des mesures similaires aux mesures conventionnelles d'évaluation du classement des documents à savoir la **précision** et le **rappel**, mais en tenant compte des scores des serveurs calculés par la méthode de classement de serveur à évaluer. Pour chaque x (le nombre de serveurs sélectionnés) entre 1 et $|S''|$, deux mesures P_x et R_x sont calculées :

$$P_x = \frac{|\{S''_i \mid (1 \leq i \leq x) \bigwedge (Score(S''_i) > 0)\}|}{x}$$

$$R_x = \begin{cases} \frac{\sum_{i=1}^{i=x} Score(S''_i)}{\sum_{i=1}^{i=x} ScoreI(S''_i)} & \text{si } \sum_{i=1}^{i=x} ScoreI(S''_i) > 0 \\ 1 & \text{sinon} \end{cases}$$

où $Score(S''_i)$ est le score du serveur S''_i, $ScoreI(S''_i)$ son score idéal pour la requête Q. L'inconvénient de cette approche est qu'elle ne tient pas compte du nombre de documents pertinents dans un serveur. Ainsi, l'inversion de rang entre les serveurs ne contenant pas le même nombre de documents pertinent n'est pas pénalisé.

3. Lu et al. [LCC96] proposent des mesures qui prennent en compte le nombre de documents pertinents contenus dans les serveurs. Ainsi, pour chaque x (le nombre de serveurs sélectionnés) entre 1 et $|S''|$, les auteurs proposent les formules suivantes :

$$P_x = \frac{\sum_{i=1}^{x} r_i}{x}$$

$$R_x = \frac{\sum_{i=1}^{x} r_i}{|Pert|}$$

où : r_i est le nombre des documents pertinents contenus dans le serveur S''_i ;
$|Pert|$ est le nombre total des documents pertinents contenus dans l'ensemble des éléments de S.

2.4 Méthodes de sélection de serveurs

La sélection de serveurs a pour intérêt majeur de réduire le nombre de serveurs à sélectionner tout en ne sacrifiant rien (ou peu) de la performance atteinte par l'interrogation de tous les serveurs. Nous passons en revue dans cette section, les méthodes proposées dans la littérature et nous dressons dans la section suivante une comparaison de ces méthodes.

2.4.1 Catalogue des méthodes de sélection de serveurs

2.4.1.1 L'approche naïve

Il est sans doute plus simple d'ignorer l'étape de sélection de serveur et d'envoyer la requête à tous les serveurs connus du système. D'une part, ce choix, souvent adopté par les usagers débutants, s'avère très onéreux en termes de ressources et de communication, ce qui génère des temps d'attente important. D'autre part, interroger systématiquement tous les serveurs risque de dégrader la qualité de recherche du système. En effet, interroger un serveur qui ne contient pas de document pertinent augmente le bruit dans la réponse. Effectuer la sélection revient donc à diminuer le coût de la recherche et augmenter, éventuellement, la performance du système.

2.4.1.2 La sélection manuelle

L'utilisateur se voit confié la tâche de choisir les serveurs sur lesquels il voudra effectuer sa recherche. Par exemple, dans West-Law[1], l'utilisateur sélectionne les sources de droit qu'il désire consulter. Cette sélection se fera sur la base des descriptions des serveurs ou sur des connaissances *a priori* qui, en général, sont très restreintes.

2.4.1.3 *CORI*

Callan et al. [CLC95] ont introduit un SRID appelé COllection Retrieval Inference network (*CORI*). Le principe est de considérer une collection comme un immense document virtuel et d'effectuer le classement des serveurs de manière similaire au classement des documents. Chaque serveur S_i est représenté par la fréquence document de chaque terme t_j de sa collection. Une autre information utilisée pour calculer le score d'un serveur pour un requête donnée est la fréquence collection pour chaque terme de la requête. Cette méthode est détaillée dans le chapitre 4 où nous allons la comparer à nos approches de sélection.

1. Service en ligne pour la recherche dans le domaine juridique : http://www.westlaw.com

2.4.1.4 CVV

Yuwono et al. [YL97] présentent un SRID appelé D-WISE dont la partie sélection de serveurs est nommée sélection par CVV (Cue Validity Variance) car elle est basée sur la CVV des termes de la requête. Pour une collection donnée, la CV (Cue Validity) d'un terme indique sa capacité à distinguer les documents entre eux. Notons que cette valeur est similaire à l'Idf classique sauf que Idf porte sur le pouvoir d'un terme à distinguer les documents où il apparaît des autres documents d'une collection. La CVV est la variance de CV à travers toutes les collections du système.
Un serveur S_i est représenté au niveau du courtier par le nombre de documents que sa collection contient, et la fréquence documents de chaque terme qu'il contient. Le score d'un serveur est obtenu en combinant les CVV des termes de la requête et leurs fréquences-documents. Le score d'un serveur fournit des indications sur la concentration des termes de la requête dans la collection du serveur.

2.4.1.5 Gloss

Gravano et al. [GGMA99] ont introduit Gloss (Glossary of Servers Server). La première version de ce système fonctionne dans un environnement booléen où chaque serveur utilise un SRI booléen. Gloss utilise la fréquence documents et la taille des collections pour calculer le score d'un serveur. Ce dernier est le nombre estimé de documents pertinents que contient le serveur.

Gloss a été généralisé par la suite à un environnement vectoriel où chaque serveur utilise un SRI vectoriel, cette nouvelle version de Gloss est appelée vGloss. On suppose dans ce cas que le même SRI est utilisé par tous les serveurs. Pour une requête donnée, le score d'un serveur est la somme des scores de ses documents supérieurs à un seuil donné. Bien sûr, ces scores n'étant pas connus par le courtier, ils sont estimés. L'estimation des scores des documents est basée sur les informations que le courtier détient concernant les collections. Les informations concernant un serveur sont représentées par deux vecteurs, celui des valeurs de fréquence documents pour chaque terme de la collection et celui de la somme des poids de chaque terme dans les documents de la collection.

2.4.1.6 Sélection probabiliste

Baumgarten [Bau99a, Bau99b] introduit un SRID basé sur un modèle probabiliste. La méthode consiste à estimer à partir des statistiques, la distribution des RSVs (*Retrieval Status Value*) à travers les serveurs. Le RSV d'un document est l'estimation de la pertinence de ce document par rapport à la requête. Le but de la sélection dans cette approche n'est pas d'estimer la distribution des documents pertinents, mais de détecter les collections qui permettent d'obtenir les l documents les mieux classés si toutes les collections sont sélectionnées.

2.4.1.7 Lightweight probes (LWP)

Hawking et al. [HT99] proposent d'effectuer la sélection de serveurs en utilisant des requêtes de sonde (*probe queries*). Au moment de l'interrogation, le courtier envoie dans un premier temps une requête de sonde à tous les serveurs. Une requête de sonde est une requête de taille réduite construite en gardant p termes de la requête initiale. La méthode suppose la coopération active des serveurs sous-jacents. En effet, ces derniers doivent répondre à la requête de sonde par des informations telles que le nombre total des documents que contient le serveur, le nombre de documents contenant les termes de la requête de sonde, des informations sur la proximité et la co-occurrence des termes de la requête de sonde. Ces informations servent à calculer le score de chaque serveur.

2.4.1.8 Sélection par apprentissage

Voorhees [Voo95] propose une approche qu'elle considère comme une approche de fusion de résultats, nous la considérons à mi-chemin entre une méthode de fusion et une méthode de sélection de serveurs. La méthode consiste à estimer λ_i, le nombre de documents à retrouver par le serveur S_i, étant donné un nombre N de documents total à chercher (on a donc : $\sum_{i=1}^{|S|} \lambda_i = N$). Les serveurs dont ce nombre est nul ne sont pas sélectionnés d'où le fait de considérer cette approche comme une méthode de sélection. Le but est de choisir les λ_i de telle façon qu'on puisse obtenir le maximum de documents pertinents dans la liste finale. Pour cela, la distribution des documents pertinents dans les listes retournées par les serveurs est estimée à l'aide d'un processus d'apprentissage. Deux alternatives pour l'apprentissage sont proposées :

1. *Modelling Relevant Document Distribution (MRDD).* En partant d'un ensemble Q de requêtes qu'on soumet aux serveurs on définit pour chaque requête q_j dans Q, la distribution A_{ij} des documents pertinents dans la liste des documents reçue en interrogeant un serveur S_i. La distribution A_{ij} est représentée par un vecteur contenant les positions des documents pertinents dans la liste de réponses du serveur S_i à la requête q_j.
   Pour chaque nouvelle requête q soumise au système, on calcule la distance entre cette requête et les requêtes de Q. Pour ce faire, les requêtes sont représentées par des vecteurs de fréquences des termes. La distance entre deux requêtes est le cosinus des deux vecteurs qui les représentent. À l'aide des distances calculées, les k requêtes les plus proches de q sont déterminées. La moyenne des distributions associées à ces k requêtes est attribuée à q. À partir de cette moyenne, et à l'aide d'une fonction de maximisation, λ_i est calculé pour chaque serveur.

2. *Query Clustering (QC).* Une autre solution est de découper l'ensemble des requêtes d'apprentissage en plusieurs groupes de requêtes. Le critère de groupement est le nombre de documents pertinents en commun. L'hypothèse sur laquelle repose cette approche est la suivante : si deux requêtes

retrouvent un nombre important de documents en commun, alors elles ont le même thème. Ce thème est représenté par le vecteur centroïde des requêtes (une requête est un vecteur) du même groupe.
Le groupement des requêtes est effectué pour chaque serveur. À chaque centroïde est associé un poids qui représente la moyenne des scores des documents pertinents retrouvés pour les requêtes du groupe sur le serveur interrogé.

À la réception d'une nouvelle requête q, celle-ci est associée au centroïde le plus proche (distance calculée comme pour MRDD par le calcul du cosinus). Soit, w_i le poids de ce centroïde, λ_i est calculé comme suit :

$$\lambda_i = N \cdot \frac{w_i}{\sum_{j=1}^{|S|} w_j}$$

2.4.1.9 Sélection avec prise en compte de la performance des SRI locaux

Craswell et al. [CBH00] proposent de prendre en compte dans la sélection des serveurs, non seulement la probabilité qu'un serveur contienne des documents pertinents, mais aussi sa capacité à les retourner. En effet, il n'est pas intéressant de sélectionner un serveur qui contient des documents pertinents mais que le SRI local n'est pas en mesure de retrouver. Craswell et al. tentent de vérifier alors la proposition précédente. Ils ajoutent un paramètre de mesure de performance à la formule de calcul du score établie par *CORI*. Ce paramètre est indépendant de la requête posée. Pour calculer le paramètre de performance d'un serveur, une opération de rétro-action automatique est effectuée pour un ensemble de requêtes de sonde. La rétro-action automatique consiste à considérer comme pertinents l'ensemble L_i des dix premiers documents retournés par le SRID à une requête de sonde q_i. Ainsi, la valeur du paramètre de performance d'un serveur est la moyenne, pour toutes les requêtes de sondes, des proportions de ses documents dans Li.

2.4.1.10 Pharos

Dolin et al. [DAEA96] conçoivent un SRID basé sur le concept de *hiérarchie d'information*. Une *hiérarchie d'information* est un arbre qui classifie l'information par date, thème, région géographique, etc. Par exemple, l'arbre qui classifie l'information géographique peut contenir un nœud *U.S.A.*, ce nœud aura plusieurs fils dont *Californie* et *Floride*, le nœud *Californie* aura à son tour des fils comme, par exemple, *Santa Barbara*, etc.
Une *taxonomie d'information* est associée à chaque *hiérarchie d'information*. Une *taxonomie d'information* est une structure d'arbre où les documents peuvent être classifiés selon leur contenu. Le nœud de chaque arbre est formé de deux champs : un label (terme ou phrase) provenant du nœud correspondant dans la *hiérarchie d'information* et une valeur numérique représentant la proportion de

documents dont le contenu correspond au label.
Chaque serveur classe ses documents selon les différentes *taxonomies* de Pharos. Pharos utilise deux types de courtiers pour la sélection de serveurs. Les courtiers de haut niveau et les courtiers de niveau intermédiaire. Les courtiers de haut niveau contiennent tous les mêmes données sur le contenu des collections. Ces données sont limitées aux plus hauts niveaux des *taxonomies d'information.* Chaque courtier de niveau intermédiaire contient des informations associées à une *hiérarchie d'information.* Les courtiers de niveau intermédiaire contiennent plus de détails sur les collections qu'ils indexent, la partie de la *taxonomies d'information* associée qui n'est pas mise dans les courtier de haut niveau.
Le processus de sélection se fait en deux phases, les courtiers de haut niveau sélectionnent les courtiers du niveau intermédiaire susceptibles de conduire à des serveurs pertinents. Ensuite, les courtiers sélectionnés vont à leur tour sélectionner les serveurs jugés pertinents.

2.4.1.11 Autres méthodes

Savoy et Rasolofo [SR00] proposent une sélection de serveurs basée sur une méthode d'apprentissage utilisant un arbre de décision répondant à la question : est-ce qu'un serveur donné peut répondre d'une manière satisfaisante à la requête courante?
Le processus d'apprentissage consiste à fournir à la machine, pour chaque serveur, et pour chaque requête d'apprentissage, un ensemble de couples (attribut, valeur) et la décision prise, dans ce cas, sur la sélection ou non du serveur en question.
Plusieurs attributs ont été pris en compte, par exemple, le nombre de termes de la requête, la fréquence documents des premiers termes de la requête, la moyenne de la fréquence documents des termes de la requête, etc.
Le traitement d'une nouvelle requête q consiste à trouver par la méthode des k plus proches voisins [MST94] les trois requêtes d'apprentissage les plus proches de q. Ensuite, la décision prise pour la majorité de ces trois requêtes est adoptée pour q.

Zobel [Zob97] propose et compare plusieurs techniques de calcul des scores des serveurs basées sur des informations telles que le nombre de collections, le nombre total de documents contenus dans les collections, le nombre de documents de chaque collection et pour chaque terme de la requête : la fréquence document dans chaque collection, la fréquence d'occurrence de ce terme dans une collection. Le calcul du score d'un serveur peut se faire selon un ou plusieurs des critères suivants : l'occurrence des termes de la requête dans la collection du serveur, la fréquence de ce terme dans la collection, la proportion de documents contenant les termes de la requête dans la collection, etc.

Xu et Croft proposent deux manières d'augmenter la performance de *CORI*. La première consiste à enrichir la requête avant d'effectuer la sélection par *CORI* en utilisant une méthode d'expansion des requêtes appelée *local context analysis*

[XC96]. La seconde consiste à enrichir les représentants des serveurs avec des informations sur les phrases qu'ils contiennent [XC98].

Xu et Croft [XC99] étudient l'impact, dans la RID, du regroupement des documents selon leur thème. Pour ce faire, l'algorithme de groupement K-means est utilisé. Cet algorithme consiste à considérer les k premiers documents d'une collection comme un groupe, chacun des documents restants est associé à un groupe si son contenu est proche de celui du groupe. Cette proximité est calculée par l'algorithme de Kullback-Leibler.
Chaque groupe est alors indexé. Pour chaque groupe on attribue un vecteur des fréquences des termes dans ses documents.
À la réception d'une requête, la mesure de divergence de Kullback-Leibler est utilisée pour calculer la pertinence d'un groupe à une requête. Une fois cette pertinence établie, la requête est dirigée vers les collections qui contiennent les documents du groupe.
Le groupement est effectué soit localement au niveau des serveurs, soit globalement au niveau du courtier. Dans le premier cas, les index des groupes sont envoyés au courtier. Dans le deuxième cas, il est nécessaire de ramener la totalité des documents des collections au niveau du courtier.

Moffat et Zobel [MZ95] expérimentent la sélection de subdivisions de collections. Chaque collection est subdivisée en plusieurs blocs de B documents. Au niveau du courtier, ce sont les représentants des blocs qui sont sauvegardés au lieu des représentants des collections. L'intérêt de cette approche est de restreindre la recherche à des portions de collections les plus susceptibles de contenir des documents pertinents.

NetSerf [CH95] est un système dans lequel les descriptions des collections sont faites manuellement. Au moment de l'interrogation, le courtier effectue la correspondance entre d'une part, les représentations sémantiques de la requête et d'autre part, celles des descriptions des collections. Ces représentations sont construite à l'aide d'un thésaurus sémantique (WordNet) et d'un dictionnaire en ligne (Webster's Dictionnary) qui sont utilisés aussi pour étendre la requête et essayer de la désambiguïser.

SavySearch [DH97] est un courtier disponible sur le Web dont le fonctionnement est basé sur les deux états "consulté" et "sans résultats" d'un serveur. Dans ce cas, le score des serveurs consultés par l'utilisateur pour une requête contenant un terme t_k seront augmentés lors d'une sélection future pour une requête contenant également le terme t_k. De la même façon, les scores des serveurs qui ne répondent pas à une requête contenant t_k seront diminués pour les nouvelles requêtes avec t_k.

Profusion [FG99] est également un courtier sur le Web dont le processus de sélection de serveurs repose sur les visites de l'utilisateur à ces serveurs.

Fuhr [Fuh99] propose un modèle dont le coût de la recherche est l'un des critères

de sélection. Pour un nombre N de documents pertinents désirés, le modèle :

1. calcule une estimation du nombre de documents pertinents dans chaque serveur ;
2. calcule le coût d'interrogation de chaque serveur en fonction du nombre de documents que le courtier lui demande de retourner. Ce nombre est déterminé en utilisant une estimation des courbes de précision/rappel du serveur ;
3. combine les coûts précédents en une fonction de coût global. La minimisation de cette dernière fonction indique le nombre optimal r_i de documents à demander au serveur S_i. Implicitement, chaque r_i non nul indique que le serveur S_i est sélectionné.

Wais [KM91] est un système dans lequel les serveurs sont classifiés et résumés manuellement, la sélection des serveurs se fait par rapport à la présence des termes de la requête dans les dénominations des catégories ou dans les résumés des serveurs.

2.4.2 Étude comparative

2.4.2.1 Comparaison des performances

Concernant les performances des méthodes retrouvées dans l'état de l'art, il est difficile de les comparer directement. En effet, les évaluations ont été réalisées dans des environnements différents et avec des mesures d'évaluation parfois différentes. La table 2.1 montre les environnements de test rencontrés à travers les études expérimentales de l'état de l'art et indique la variété des environnements évalués. Les notations utilisées dans la table sont les suivantes : TREC : Text REtrieval Conference ; TREC-x : corpus de la x-ième conférence de TREC ; CACM, CISI, MED : corpus constitués et distribués par Gérald Salton et ses étudiants ; CRAN : Cranfield collection.

TAB. 2.1 – *Une sélection d'environnements de test utilisés dans l'état de l'art.*

Référence	Collection test	Nombre de collections	Numéros de requêtes (pour TREC)	Mesure
[CLC95]	TREC-1	17	51-150	*EvalPerf* et *EvalSel*
[CBH00]	TREC-8	956	401-450	*EvalPerf*
[HT99]	TREC-5	98	251-300	*EvalPerf* et *EvalSel*
[SR00]	TREC-8	4	401-450	*EvalPerf*
[XC99]	TREC-3	100		*EvalPerf*
[VT97]	TREC-1, TREC-2 et TREC-3	5	51-200	*EvalPerf*
[GGM95]	groupes de news	53	6800 usagers	*EvalSel*
[PFC00]	TREC-4	100, 236	51-150	*EvalPerf*
[Zob97]	TREC-2 TREC-3	43 91	51-150 202-250	*EvalSel*
[YL97]	CACM, CISI, CRAN, MED	4	431 requêtes	*EvalSel*
[MZ95]	TREC-1	9	51-150	*EvalPerf*

Dans ce qui suit, nous décrivons quelques études expérimentales publiées.

Callan, Lu et Croft [CLC95] utilisent *EvalPerf* afin d'évaluer *CORI* sur le corpus TREC-1 divisé en 19 collections équivalentes en nombre de documents. Cependant, aucune comparaison de *CORI* à d'autres approches n'a été effectuée.

Craswell, Bailey et Hawking [CBH00] comparent plusieurs méthodes de sélection de serveurs entre autres *CORI*, vGloss, CVV et *CORI*-enrichi (voir 2.4.1.9). Le corpus utilisé est composé de 956 collections obtenues par un découpage par source à partir des documents de TREC8 [HVCB99b]. L'étude expérimentale montre que lorsque l'on sélectionne dix serveurs sur les 956, *CORI* dépasse vGloss qui surpasse CVV. *CORI*-enrichi en utilisant la rétro-action automatique[2] ne présente pas une amélioration significative néanmoins lorsque les informations sur la performance des serveurs sont manuellement calculées, i.e. la rétro-action n'est pas automatique[3], l'amélioration des résultats est importante. Ceci renforce l'hypothèse que *sélectionner les serveurs qui ont plus de chance de retourner des documents pertinents est plus intéressante que sélectionner les serveurs qui sont susceptibles de contenir des documents pertinents.*

Hawking et Thislewaite [HT99] effectuent *EvalPerf* et *EvalSel* sur l'ensemble des documents de TREC5 divisé d'abord selon la source des documents (division en six), ensuite en 98 sous-collections telles que ces dernières aient approximativement la même taille (en octet). La taille des collections varient de 1548 documents à 8527. Les tests concernent des méthodes de sélection de serveurs qui ne nécessitent pas de descriptions des serveurs ou des informations globales sur les serveurs, entre autres LWP et celle de Voorhees. LWP s'est avérée être la plus performante des deux point de vues : *EvalPerf* et *EvalSel.*

Savoy et Rasolofo [SR00] présentent les résultats de leur approche de sélection en variant les SRI locaux, et les comparent à *CORI* qui, dans ce contexte (4 collections), s'avère moins performante.

Xu et Croft dans [XC99] ont comparé *CORI* à leur méthode de sélection basée sur le regroupement des documents par thèmes (voir 2.4.1.11). Sur un corpus de test constitué de 100 collections (subdivision des données de TREC), l'évaluation *EvalPerf* a montré la performance de l'approche de sélection par rapport à CORI.

Voorhees et Tong dans [VT97] ont évalué leur deux méthodes de sélection de serveurs à savoir QC et MRDD (voir 2.4.1.8) en utilisant *EvalPerf.* Les deux méthodes ont été combinées à la méthode de fusion proposée par Voorhees [Voo95]. Les expérimentations ont été conduites sur quatre collections de TREC. Sur les 150 requêtes de test utilisées, la partie apprentissage a été effectuée sur 100 requêtes et la partie test de performance sur les 50 requêtes restantes. Les

2. La rétro-action automatique consiste à considérer, dans ce cas, les dix premiers documents retournés par le SRID pertinents.

3. La rétro-action non automatique consiste à attribuer un jugement humain à la pertinence des dix premiers documents retournés par le SRID

résultats des expérimentations montrent qu'au niveau de la performance QC et MRDD sont inférieures à l'approche centralisée.

Gravano et Garcia-Molina dans [GGM95] évaluent la sélection de vGloss par P_x et R_x (voir 2.3.2) et montrent que vGloss donne de bons résultats comparé au classement idéal. Le classement idéal, dans ce cas, consiste à classer les serveurs selon la somme des scores de leurs documents dont le score est supérieur à un certain seuil.

Powell et al. [PFC00] vérifient la consistance de l'hypothèse suivante : si une méthode de sélection de serveurs efficace est effectuée alors les résultats d'un SRID peuvent être plus performants que les SRIC, et cela dans différents environnements. Pour cela, Powell et al. ont effectué des expérimentations dans trois environnements différents obtenus à partir des données de TREC4 [Har96]. Le premier environnement est formé de 100 collections d'approximativement 30 méga-octets (MO), dont les documents de chacune proviennent de la même source. Le deuxième est formé de 236 collections composées chacune de documents provenant de la même source, dont le mois et l'année de publication sont les mêmes. Enfin, le troisième environnement est constitué de 236 collections dont la taille de chacune est d'approximativement 2900 documents d'une même source.
Les expérimentations, basées sur l'évaluation *EvalPerf*, ont permis de confirmer l'hypothèse citée au-dessus.

Yumono et Lee [YL97] ont effectué *EvalSel* (P_x et R_x) sur plusieurs approches, entre autres CVV, vGloss, bGloss et *CORI*. Les résultats montrent que CVV arrive en premier, vient ensuite vGloss, et enfin *CORI* et bGloss arrivent en dernier. Leurs tests se sont effectués sur les corpus CACM, CISI, CRAN et MED. Cependant, ces collections sont trop petites (7 MO en total, et 7097 documents), les résultats ne sont donc pas convaincants.

Zobel [Zob97] présente une évaluation *EvalSel* pour plusieurs approches utilisant des informations statistiques pour calculer le score des serveurs (voir 2.4.1.11). Zobel utilise deux corpus de TREC (TREC-2 et TREC-3) découpés respectivement en 43 et 91 collections. La subdivision de TREC-2 a la particularité que les documents d'une même collection appartiennent à la même source. La subdivision de TREC-3 est aléatoire.
Les différentes approches de classement de serveurs, testées dans ces travaux, sont comparées à l'approche idéale, celle qui classe les serveurs selon le nombre de document pertinents qu'ils contiennent. Les résultats montrent que les approches testées (approches utilisant des informations statistiques pour calculer le score des serveurs) ne sont pas très différentes. L'approche se distinguant légèrement est celle utilisant le produit interne entre les deux vecteurs associés respectivement à la requête et à la collection. Le vecteur de la requête est composé des poids des termes de la requête, et le vecteur de la collection est composé des fréquences-document des termes apparaissant dans la collection.

Moffat et al. [MZ95] effectuent une évaluation *EvalPerf* de leur approche pour

différentes valeurs de la taille des blocs qu'ils indexent (voir 2.4.1.11). La seule comparaison présentée est faite par rapport à l'approche centralisée. Les résultats ont montré que le seul cas où l'approche est bénéfique est celui de la subdivision en blocs de dix documents. Or, ce cas a l'inconvénient, qu'au niveau du courtier, l'index des blocs est de taille trop importante.

Sur la base des études expérimentales précédentes, il s'avère difficile de tirer des conclusions définitives sur la performance d'une méthode par rapport à toutes les autres.

Nous verrons dans la section suivante qu'il y a d'autres critères sur lesquels les méthodes peuvent être comparées.

2.4.2.2 Comparaison selon différents autres critères

Nous résumons dans la table 2.2 page 48 les caractéristiques des différentes méthodes citées dans la section 2.4.1. Nous présentons d'abord ici les différents champs de la table de comparaison.

1. Le but de la méthode (But Meth.) : les méthodes de sélection diffèrent selon le but qu'elles veulent atteindre, à savoir, détecter :
   (a) les serveurs où il y a plus de chance de trouver des documents pertinents(cDp). Le nombre des documents pertinents n'est pas important dans ce cas ;
   (b) les serveurs qui contiennent un grand nombre de documents pertinents(nDp) ;
   (c) les serveurs qui ont plus de chance de retourner des documents pertinents($crDp$). L'intérêt dans ce cas n'est pas seulement de retrouver des serveurs qui contiennent des documents pertinents, mais ceux qui sont capables de les retourner ;
   (d) les serveurs qui sont le moins coûteux à interroger (en temps ou en ressources)(Smc).
2. La nature du représentant d'un serveur pour une méthode (Nat. Repr.) : les représentants des serveurs peuvent être de plusieurs formes :
   (a) des informations statistiques (Is) concernant les termes des collections comme, par exemple, la fréquence, la fréquence documents, la taille en documents d'une collection, etc. ;
   (b) des descriptions textuelles (Dt) des contenus des collections comme, par exemple, un ensemble de mots clés ou un ensemble de paragraphes ;
   (c) des informations sur les précédentes interrogations (Dp) ;

(d) des informations pour estimer le coût de la recherche (Ic) sur un serveur, comme, par exemple, le temps d'attente, le coût de traitement de la requête, le temps de transfert d'un document du serveur, le coût de ce transfert, etc. ;

(e) des informations permettant d'avoir une idée sur la performance d'un serveur(Ip), en d'autres termes des informations qui permettent de prédire le degré de capacité d'un serveur à retrouver des documents pertinents.

3. Le mode d'acquisition du représentant d'un serveur (Acq. Repr.) : Il y a trois possibilités d'acquérir les représentants des serveurs :

(a) par la coopération active (CA) des serveurs : les serveurs fournissent toutes les informations demandées par le courtier, et, dans certains cas, changent leur fonctionnement interne (indexation, pondération) pour s'adapter à la demande du courtier ;

(b) par la coopération passive (CP) des serveurs : les serveurs n'émettent pas d'objections à ce que le courtier les interrogent intensément ;

(c) sans aucune coopération des serveurs (SC) : il n'y aucune contrainte sur les serveurs. Ce cas n'apparaît pas dans notre table, car aucune des méthodes citées ne répond à ce critère.

4. La maintenance (Maint.) : La maintenance fait référence à la fréquence des mises à jour nécessaires des représentants des serveurs afin de maintenir la représentativité des serveurs. Cette maintenance dépend de la durée de vie des représentants ainsi que de leur nature. Ainsi, il faudra une maintenance :

(a) régulière (R) pour les méthodes dont le représentant d'un serveur est permanent dans le courtier ;

(b) minime (M) pour les méthodes dont le représentant d'un serveur ne dépend pas du contenu des documents individuels ;

(c) inutile (I) pour les méthodes dont le représentant d'un serveur est ponctuel (construit au moment de l'interrogation).

5. L'extensibilité (Extens.) : L'extensibilité d'une approche est le pouvoir d'une méthode à s'adapter à l'expansion de ses données. Ainsi l'extensibilité peut être mesurée par la taille du représentant d'un serveur. Plus cette taille est importante moins l'approche est extensible. Ce critère est important dans le sens où Internet ne cesse de s'agrandir de jour en jour ce qui exige de développer des méthodes très extensibles. Nous définissons donc par :

(a) E les méthodes extensibles ;

(b) M les méthodes moyennement extensibles ;

(c) N les méthodes très peu extensibles.

TAB. 2.2 – *Table de comparaison des méthodes de sélection de serveurs*

Méthodes	But Meth.	Nat. Repr.	Acq. Repr.	Maint.	Extens.
Naïve	-	-	-	I	E
Manuelle	-	Dt	CP	M	E
CORI	cDp	Is	CA	R	M
CVV	cDp	Is	CA	R	M
Gloss	nDp	Is	CA	R	M
Baumg	cDp	Is	CA	R	M
LWP	nDp	Is	CA	I	E
Voor	nDp	Dp	CP	R	N
Cras	crDp	Is	CA	R	M
Pharos	cDp	Is	CA	M	E
Zobel	cDp	Is	CA	R	M
Xu [XC99]	cDp	Is+Dt	CA	R	E
Moffat	cDp	Is	CA	R	N
NetSerf	cDp	Dt	CP	M	E
SavySearch	cDp	Ip	CP	I	E
Profusion	cDp	Ip	CP	I	E
Fuhr	Smc	Ic(+Is)	CA	R	N
WAIS	cDp	Dt	CP	M	E

2.5 Méthodes de fusion

La fusion des résultats provenant de différents serveurs a pour but d'offrir la transparence de la recherche sur de multiples serveurs à l'utilisateur. Pour cela, le courtier produit une seule liste de documents classés selon leur degré de pertinence à la requête. Nous présentons dans cette section un état de l'art des méthodes de fusion proposées classées selon les informations dont le courtier a besoin pour effectuer la fusion (rang, score, etc.). Pour plus de simplicité, dans ce qui suit, le terme "document" fait référence à un lien vers un document.

2.5.1 Les méthodes ne nécessitant aucune information à propos des documents

Dans un environnement où les serveurs ne retournent que des listes non triées de documents, il existent deux possibilités pour effectuer la fusion :

1. l'union des résultats : dans ce cas, le classement peut s'effectuer au hasard, par ordre d'arrivée des listes, par longueur des listes, etc.

2. récupérer les corps des documents, leur attribuer des scores à partir de leur analyse et enfin les classer. Cette démarche est adoptée par le méta-moteur Inquirus [LG98].

2.5.2 Les méthodes utilisant le rang des documents

Lorsque seul le rang d'un document est l'information disponible dans les listes de résultats, situation très fréquente sur le Web, les méthodes de fusion suivantes sont envisageables :

1. l'approche *round robin* ou *à chacun son tour* qui consiste à intercaler les documents du même rang, mais retournés par des serveurs différents. Ainsi, la liste finale sera construite de l'ensemble des documents en tête des listes à fusionner, suivi de l'ensemble des documents du deuxième rang et ainsi de suite.
2. Yuwono et al. [YL97] proposent d'utiliser le score d'un serveur, score calculé pendant la phase de sélection de serveurs, pour estimer la distance entre les documents retournés par le serveur et cette valeur estimée sera combinée au rang d'un document pour calculer son score ;
3. Voorhees [Voo95] place, à la position actuelle dans la liste finale, le document du serveur dont le score temporaire est le plus élevé. Le score temporaire d'un serveur étant le nombre de documents qui restent dans sa liste de réponse. Une fois qu'un document d'un serveur est inclus dans la liste finale, le score temporaire de ce serveur est diminué. Le processus continue jusqu'à ce qu'il n'y ait plus de documents dans toutes les listes à fusionner.

2.5.3 Les méthodes utilisant les scores locaux

Lorsque les scores des documents sont comparables, la méthode la plus évidente est sans aucun doute la *fusion par le score* appelée aussi dans la littérature anglophone *Raw Score merging* (*RSM*) consistant à classer les documents selon leur score local.
Cependant, les scores locaux sont très souvent peu comparables, même si la même méthode de recherche est utilisée (la même fonction de correspondance, la même lemmatisation, la même liste de mots vides) par tous les serveurs car le score d'un document est souvent calculé à partir des statistiques propres à la collection à laquelle il appartient. Lu et al. [LCC96] proposent, afin de diminuer l'impact de la non-comparabilité des scores sur le résultat final, de pondérer le score local d'un document par le score du serveur qui le retourne. Cette pondération se fait au niveau du courtier à la réception des résultats provenant des serveurs.

Dans un environnement où la coopération active est possible, des approches diverses ont été proposées pour pallier le problème de la non-comparabilité des scores, en exigeant évidemment que la même méthode de recherche soit utilisée :

1. Callan et al. [CLC95] ainsi que Kretser et al. [DKMSZ98] proposent de collecter des statistiques globales sur les collections, et au moment de

l'interrogation, fournir ces statistiques aux serveurs qui les utiliseront pour calculer les scores des documents ;

2. Viles et al. [VF95] proposent que les serveurs s'échangent les statistiques appropriées avant l'interrogation.

2.6 Conclusion

Dans ce chapitre nous avons introduit plusieurs aspects concernant la recherche d'information distribuée. Nous avons décrit d'abord les différentes composantes fonctionnelles d'un SRID. Ensuite, nous avons détaillé les fonctions qui déterminent la performance du système à savoir la sélection de serveurs et la fusion des résultats. Nous avons en outre étudié les méthodes proposées pour effectuer chacune de ces deux fonctions. Comme cette thèse porte sur le problème de la sélection de serveurs, nous avons présenté une comparaison des méthodes de sélection de serveurs recensées dans la littérature selon d'autres critères que la qualité de réponse du système tels que son extensibilité et la facilité de sa maintenance. Nous avons également rapporté les résultats des différentes études expérimentales publiées dont le but est de comparer les performances de certaines méthodes de sélection de serveurs. Nous avons montré qu'il était difficile d'avoir une idée très claire et précise des performances de ces méthodes à cause de la diversité des environnements de tests. A travers cette étude de l'état de l'art, nous sommes arrivés aux constatations suivantes :

- la plupart des méthodes de sélection proposées nécessitent une coopération passive ou active des serveurs, ce qui n'est possible qu'à travers des accords de type commercial sur Internet ;
- la majorité des méthodes de sélection ne prennent pas en compte la qualité de réponse des systèmes de recherche locaux. Or, il n'est pas intéressant de sélectionner un serveur contenant des documents pertinents mais qui est incapable de les retourner.

Le chapitre suivant sera consacré à la présentation de notre approche de sélection. Nous avons tenu compte des constatations précédentes dans nos recherches et nous avons proposé une méthode qui ne nécessite aucune coopération des serveurs et tient compte de la qualité de réponse (performance) des moteurs locaux.

Chapitre 3

Nos approches de sélection de serveurs

Table des matières

3.1 Introduction

Nous avons présenté dans le chapitre précédent, les problématiques rencontrées dans la recherche d'information distribuée, et principalement la sélection de serveurs. Dans ce chapitre, nous proposons une méthode d'acquisition des données qui serviront à effectuer la sélection de serveurs ainsi que trois approches de sélection de serveurs.

Ce chapitre est organisé de la manière suivante. Nous présentons respectivement dans les sections 3.2 et 3.3 les buts visés et les hypothèses posées. Dans la section 3.4, nous commençons par expliquer la manière d'acquérir les informations servant à effectuer la sélection, suivie des détails sur les stratégies de sélection proposées dans cette thèse. La section 3.5 est consacrée à la présentation de nos motivations dans les choix faits. Enfin la section 3.6 fera l'objet d'une comparaison de nos approches et celles proposées par d'autres auteurs.

3.2 Objectifs

Dès le début de nos travaux de recherche, nous nous sommes fixé le but principal suivant : *concevoir une méthode qui soit la moins contraignante possible (une méthode n'exigeant ni coopération des serveurs, ni mise à jour de données) d'une part, et d'autre part, qui atteigne une performance raisonnable.*

L'étude des travaux antérieurs nous a démontré la présence de nombreuses contraintes lors de mise en place d'un SRID. Nous nous sommes alors fixé l'objectif de concevoir un système performant et reposant sur le moins de contraintes possibles telles que la coopération, la maintenabilité et l'extensibilité.

3.2.1 La performance

Nous visons dans ce travail de thèse à proposer une approche de sélection de serveur qui permette au SRID d'atteindre (voire de dépasser) la performance des SRICs. Nous entendons par *performance* la qualité des résultats qu'un système retourne à l'utilisateur.

3.2.2 La coopération

On rappelle que la coopération peut être active ou passive.

Dans le premier cas, la coopération pour les serveurs peut aller d'une simple transmission de leurs données internes, à l'adaptation de leur fonctionnement interne aux exigences du courtier. Dans le cas réel, c'est une coopération très difficile à mettre en place, sous réserve qu'une seule organisation conçoive et implémente courtier et serveurs, par exemple, une grande bibliothèque numérique.

Dans le cas de coopération passive, la coopération se résume à l'inclusion des serveurs dans le SRID, mais cette coopération n'est pas évidente pour autant comme dans le cas des SRIDs sur le Web (souvent appelés métamoteur) par exemple. En effet, le métamoteur est un concurrent potentiel aux moteurs sous-jacents, il gagne de l'argent (vente du logiciel, publicité) en utilisant les éléments appartenant aux moteurs et annuaires à leur dépens, et l'utilisation des métamoteurs évite la vision de certaines bannières de publicité des moteurs car il n'y a pas passage par leur page d'accueil.
Nous aspirons donc à éviter toute forme de coopération dans notre approche.

3.2.3 La maintenabilité

La maintenance s'avère indispensable, car les informations que le courtier détient au sujet du contenu des serveurs doivent être mises à jour régulièrement. En effet, les documents des serveurs peuvent souvent subir des modifications importantes ou même disparaître.
Notons que les SRIDs se basant sur des descriptions globales du contenu des serveurs exigent moins de maintenance que les systèmes utilisant des informations statistiques. En effet, les descriptions générales ne se réfèrent que peu au contenu de chaque document, ainsi, une modification d'un document n'altère que peu le thème général de la collection.

Les mises à jour sont coûteuses et la volatilité des documents sur le Web ne fait qu'accentuer le besoin de mises à jour. C'est pour cela que nous avons décidé de proposer une solution n'exigeant pas de mise à jour au niveau du courtier.

3.2.4 L'extensibilité

Le but principal de la RID est de résoudre le problème de l'accroissement exponentiel de la quantité d'information disponible. Il devient alors difficile, voire impossible, de construire un index unique. Une solution plus simple consiste à distribuer la recherche. Néanmoins, Internet continue d'évoluer, et il est fort probable que le nombre de serveurs augmente aussi. Mais cela soulève un problème. En effet, si un SRID englobe un nombre considérable de collections, il devient difficile de les indexer au niveau du courtier. Nous tenterons donc d'optimiser les tailles des représentants des serveurs afin d'éviter ce problème et de faciliter l'insertion ou la suppression d'un serveur au SRID.

3.3 Hypothèses

Avant de détailler nos approches, nous présentons les hypothèses sur lesquelles nos travaux reposent.

Hypothèse(1) Une bonne sélection consiste à sélectionner les serveurs retour-

nant en premier lieu des documents pertinents.

Hypothèse(2) Les premiers documents retournés par un SRI sont représentatifs du reste de la liste des résultats.

Pour appuyer notre première hypothèse, disons que sélectionner un serveur contenant des documents pertinents mais classant ces derniers relativement bas n'est que très modérément intéressant. En effet, certaines études [WSJS01] ont montré que 45% des utilisateurs des moteurs de recherche ne consultent que la première page de dix résultats, 21,2% en consulte deux et 36,1% en consulte trois. En moyenne 1,6 pages de résultats sont consultées par recherche. Une autre étude récente [SWJT01], effectuée sur des données différentes conclut aussi que les utilisateurs ne consultent qu'un nombre très restreint de pages de résultats.

Pour appuyer notre seconde hypothèse, disons qu'un moteur de recherche peut donner des résultats très satisfaisants ou complètement inadaptés, selon la requête posée. Ainsi, le meilleur moyen de vérifier l'efficacité d'un système pour une requête donnée est d'analyser les premiers documents qu'il retourne.

3.4 Notre démarche

3.4.1 Notation

- q une requête soumise par un utilisateur ;
- n le nombre de serveurs du système ;
- S l'ensemble $\{S_1, S_2, \ldots, S_n\}$ des serveurs S_i connus par le système ;
- R_i la réponse du serveur S_i. R_i est une liste finie de couples $< d_{ij}, info_{ij} >_{j \in \mathbb{N}}$ provenant de S_i, où $info_{ij}$ est un ensemble d'information relatives à d_{ij} ;
- d_{ij} est la $j^{ème}$ référence vers un document appartenant à la collection du serveur S_i et jugé pertinent par ce serveur ;
- Sc_{ij} est le score attribué par le courtier au document d_{ij} ;
- Rep_i le représentant du serveur S_i ;
- Rep l'ensemble des représentants des serveurs, $Rep = \bigcup_{i=1..n} Rep_i$;
- k paramètre de notre approche d'acquisition des représentants des serveurs. Il représente le nombre des premiers documents analysés de chaque R_i ;
- $DocChar_i$ l'ensemble des k premiers documents de R_i chargés par le courtier afin d'être analysés ;
- $DocChar$ la réunion des $DocChar_i$, $DocChar = \bigcup_{i=1..n} DocChar_i$;
- S_q l'ensemble des serveurs à sélectionner, $S_q \subseteq S$.

3.4.2 L'acquisition des données nécessaires à la sélection

Ce point marque la différence entre notre approche et la plupart de celles proposées dans la littérature. En effet, la plupart des méthodes de sélection de serveurs supposent la coopération active des serveurs pour l'acquisition des données servant à effectuer le choix des serveurs à interroger. Notre approche construit les représentants des serveurs au moment de l'interrogation, ces représentants dépendent de la requête soumise, contrairement aux SRIDs classiques.

Les grandes lignes de notre méthode sont les suivantes : À la réception de la requête q, le courtier la soumet à tous les serveurs du système sans exception. Chaque serveur traite q et renvoie une liste de résultats. Le courtier récupère à partir de ces listes les k premiers documents. Ces derniers sont chargés et analysés pour en extraire les informations jugées utiles pour calculer un score pour chaque document.

Reprenons maintenant en détail les étapes de cette première partie de notre démarche.

3.4.2.1 L'interrogation de tous les serveurs (figure 3.1(a))

Dans cette première étape correspondant à la partie (a) de la figure 3.1, le courtier achemine la requête q, soumise par l'utilisateur, vers tous les serveurs. Le courtier attend que chaque serveur S_i traite q et retourne une liste de résultats R_i.

3.4.2.2 Le téléchargement des documents en tête (figure 3.1(b))

La partie (b) de la figure 3.1 représente cette deuxième étape. Le courtier charge les documents pointés par les k premiers éléments de chaque liste R_i. Cet ensemble de documents se nommera $DocChar_i$. Chaque élément de R_i est formé de couple $< d_{ij}, info_{ij} >$. Cependant, seul d_{ij} est utilisé pour repérer les documents et les charger. L'ensemble des $DocChar_i$ forme une base de documents nommé $DocChar$ contenant $(n \times k)$ documents. Cet ensemble $DocChar$ est une donnée temporaire et n'existe que le temps de l'interrogation.
Remarque : Dans la suite de nos propos, d_{ij} fera référence au document et non pas au pointeur de document.

3.4.2.3 L'analyse des documents et l'extraction des données requises (figure 3.1(c))

Cette étape vise à analyser les documents chargés par le courtier afin d'extraire les données statistiques qui permettent de prédire la pertinence de ces docu-

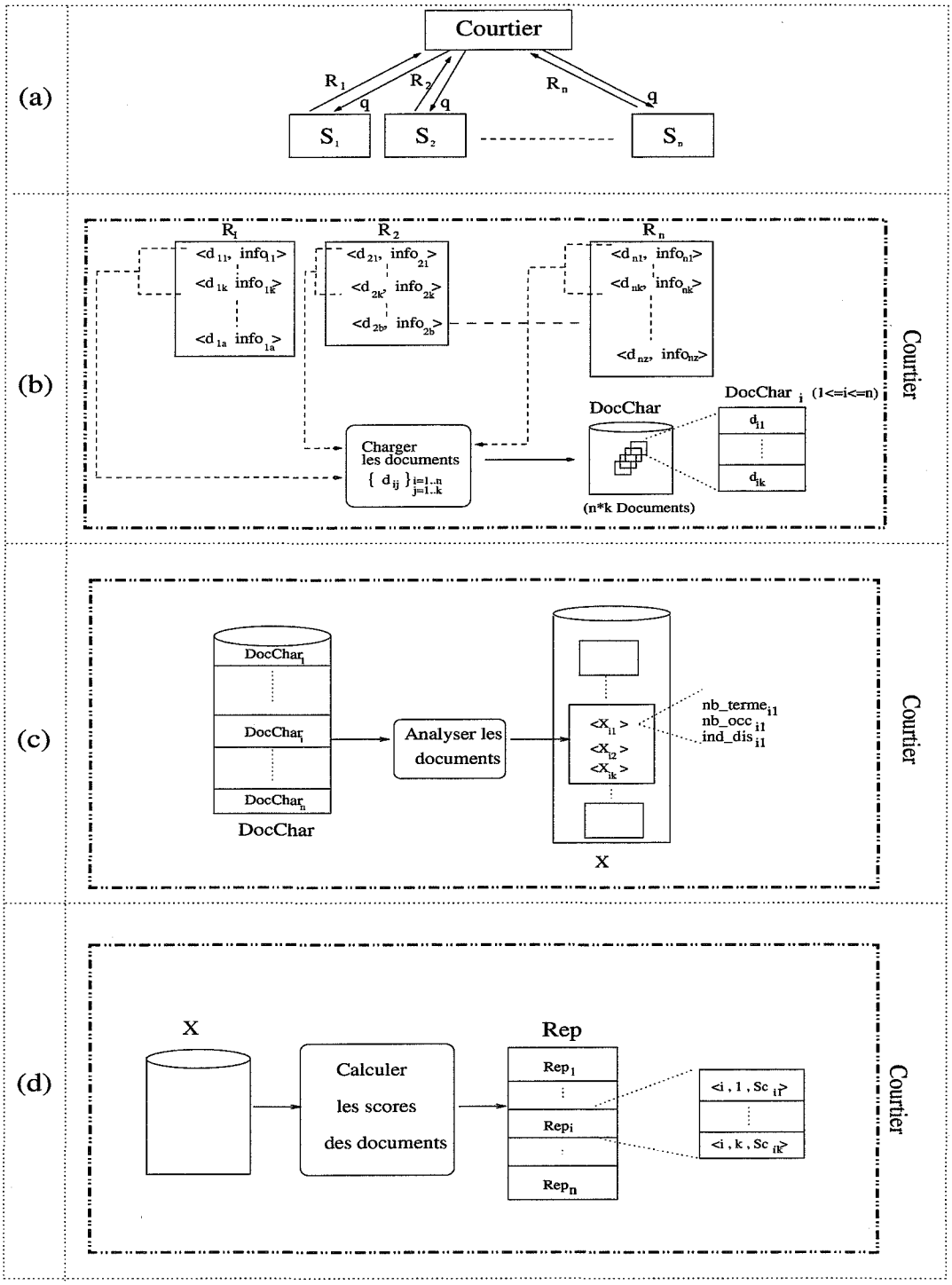


FIG. 3.1 – *Les étapes de l'acquisition de données*

ments. Nous pensons que les données susceptibles de déterminer la pertinence d'un document donné sont les suivantes :

- l'apparition des termes de la requête dans le document,
- la fréquence d'occurrence de ces termes dans le document,
- et la distance qui sépare les occurrences de ces termes dans ce document.

Dans la partie (c) de la figure 3.1, tous les éléments d_{ij} sont analysés. L'analyse d'un document d_{ij} permet d'extraire l'ensemble des informations (noté X_{ij} dans la figure 3.1) suivantes :

nb_terme_{ij}	le nombre de termes de la requête apparaissant dans d_{ij} ;
nb_occ_{ij}	la somme des nombres d'occurrences des termes de la requête dans d_{ij} ;
$ind\_dis_{ij}(t1, t2)$	un indicateur de distance entre les deux premiers termes t_1 et t_2 de la requête.

Pour le calcul de $ind\_dis_{ij}(t1, t2)$, nous adoptons la méthode suggérée par Clarke et al. [CCB95] :

$$ind\_dis_{ij}(t1, t2) = \sum_{b \in BM_{ij}(t_1, t_2)} dis_b$$

où $BM_{ij}(t_1, t_2)$ est l'ensemble des blocs, i.e. des suites de termes, minimaux de d_{ij}, délimités par les deux termes t_1 et t_2. Un bloc minimal étant un bloc ne contenant pas un autre bloc. Enfin, dis_b est l'indicateur de distance du bloc b, il est calculé comme suit :

$$dis_b = \begin{cases} \frac{A}{|b|}, & \text{si } |b| > A \\ 1 & \text{si } |b| \leq A \end{cases} \tag{3.1}$$

dis_b est donc une valeur entre 0 et 1. La constante A détermine la distance tolérée entre les termes t_1 et t_2. Par exemple, pour $A = 0$, les blocs les plus pertinents (dont le score sera égal à 1) sont ceux où les deux termes de la requête ne sont séparés par aucun autre terme. |E| est le cardinal de l'ensemble E.

Exemple de calcul :
Soit la requête et le document suivants :

q	mariage mixte
d_{ij}	Le$_1$ **mariage$_2$ mixte$_3$** :$_4$ **Mixte$_5$** ,$_6$ tous$_7$ les$_8$ **mariages$_9$** ne$_{10}$ le$_{11}$ sont$_{12}$ -$_{13}$ ils$_{14}$ pas$_{15}$ puisque$_{16}$ par$_{17}$ définition$_{18}$,$_{19}$ ils$_{20}$ unissent$_{21}$ deux$_{22}$ personnes$_{23}$ de$_{24}$ sexe$_{25}$ différent$_{26}$?$_{27}$ les$_{28}$ **mariages$_{29}$ mixtes$_{30}$** ,$_{31}$ au$_{32}$ sens$_{33}$ démographique$_{34}$,$_{35}$ concernent$_{36}$ aujourd'hui$_{37}$ environ$_{38}$ 10%$_{39}$ des$_{40}$ **mariages$_{41}$**

- Les deux termes de la requête sont contenus dans d_{ij}, donc :

$$nb\_terme_{ij} = 2$$

- Le terme *mariage* apparaît quatre fois (si l'on considère une méthode de lemmatisation qui supprime les terminaisons du pluriel) dans d_{ij} et le terme *mixte* trois fois, donc :

$$nb\_occ_{ij} = 4 + 3 = 7$$

- Les blocs de d_{ij} délimités par les deux termes de la requête sont : (2,3) (2,5) (2,30) (3,9) (3,29) (3,41) (5,9) (5,29) (5,41) (9,30) (29,30) (30,41). En éliminant ceux qui ne sont pas minimaux, nous obtenons :

$$BM_{ij}(mariage, mixte) = \{(2,3)\ (5,9)\ (9,30)\ (29,30)\ (30,41)\}$$

Ainsi, pour $A = 2$ on trouve :

$$ind\_dis_{ij}(mariage, mixte) =$$
$$dis(2,3) + dis(5,9) + dis(9,30) + dis(29,30) + dis(30,41) =$$
$$1 + \frac{2}{5} + \frac{2}{22} + 1 + \frac{2}{12} =$$
$$2.66$$

Dans le cas des requêtes à terme unique et selon [LG98], nous considérons $ind\_dis_{ij}(t_1)$ comme l'inverse de la distance (en nombre de termes) entre le début du document et la première occurrence de t_1 dans le document. Enfin, la valeur zéro est attribuée aux documents ne contenant aucun terme de la requête.

3.4.2.4 Le calcul du score de chaque document (figure 3.1(d))

Comme schématisé dans la partie (d) de la figure 3.1, à partir des éléments X_{ij}, le courtier calcule le score Sc_{ij} de chaque document d_{ij}. Les scores Sc_{ij} des documents de $DocChar_i$ formeront le représentant du serveur S_i noté Rep_i. Nous notons Rep, l'ensemble des représentants de tous les serveurs. Cet ensemble est formé de $n \times k$ éléments de type $(i, j, Sc_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq k}}$.
Nous nous sommes inspirés des travaux de Lawrence et al. [LG98] pour définir la formule de calcul du score qui est la suivante :

$$Sc_{ij} = (c_1 \times nb\_terme_{ij}) + (c_2 \times nb\_occ_{ij}) + (c_3 \times ind\_dis_{ij}(t_1, t_2)) \quad (3.2)$$

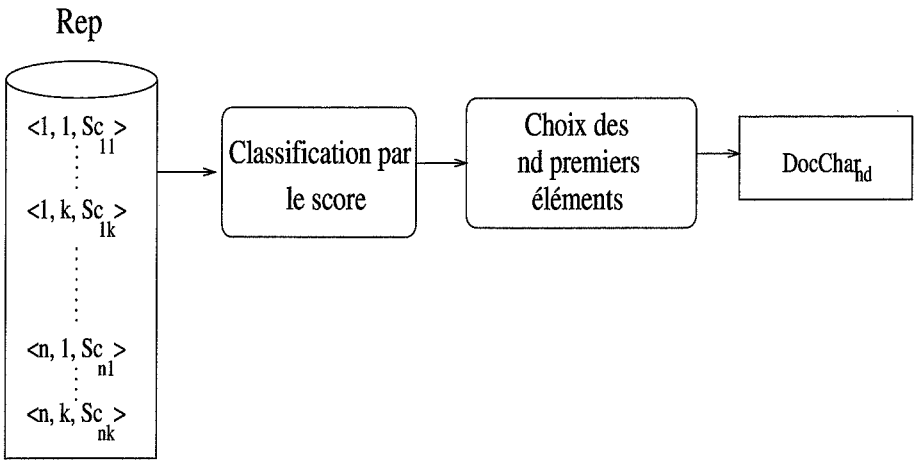


FIG. 3.2 – *La sélection des documents pertinents*

où c_1, c_2 et c_3 sont des constantes déterminant l'importance de chaque composante de la mesure de similarité.
À l'issue de cette étape, nous disposons des représentants de tous les serveurs pour la requête q et nous avons alors tous les éléments pour procéder à la sélection des serveurs susceptibles de retourner des documents pertinents.

Avant de poursuivre, une remarque s'impose. Interroger la totalité des serveurs pourrait paraître aller à l'encontre du principe même de la sélection qui vise à réduire le nombre de serveurs à interroger. Nous l'avons adoptée, car même si nous prennons le risque d'être moins efficient que d'autres approches, nous aurons atteint les objectifs cités ci-dessus. En effet, cette approche nous permet d'acquérir les représentants des collections pour une requête donnée, sans recourir à une coopération entre serveurs. Les représentants des serveurs sont créés à partir d'informations récoltées au moment de l'interrogation, et étant spécifiques à chaque requête, leur sauvegarde ne s'avère pas nécessaire. De ce fait, aucune maintenance n'est sollicitée lors d'un changement du contenu d'un serveur (ajout, modification ou suppression d'un document). De plus, l'ajout ou la suppression d'un serveur n'est qu'un simple ajout ou suppression de son adresse dans le système.

3.4.3 Les approches de calcul du score d'un serveur

Afin de pouvoir sélectionner un serveur, nous avons besoin de mesurer sa capacité à retourner des documents pertinents. Dans la littérature, cette mesure est communément représentée par un score attribué au serveur.

Pour associer un score à un serveur, nous nous basons sur la pertinence des premiers documents retournés par ce serveur. Dans ce but, nous supposons que les *nd* (paramètre constant de notre système) documents correspondants aux

scores les plus élevés (calculés selon la formule 3.2) sont pertinents. Nous appelons cette ensemble de documents $DocChar_{nd}$. Pour définir cet ensemble nous effectuons les étapes suivantes (voir figure 3.2) :

- nous classons d'abord les triplets $(i, j, Sc_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq k}}$ (éléments de Rep) selon la clé Sc_{ij} ;
- ensuite, nous sélectionnons les nd premiers triplets de cette liste triée, ces éléments correspondent donc à des triplets $(i, j, Sc_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq nd}}$, nous appelons cet ensemble Rep_{nd} ;
- enfin, nous formons $DocChar_{nd}$, l'ensemble des documents correspondant à Rep_{nd}. L'ensemble $DocChar_{nd}$ contient donc les d_{ij} tels que $(i, j, Sc_{ij}) \in Rep_{nd}$.

Pour calculer le score d'un serveur, nous avons plusieurs alternatives basée chacune sur l'une des hypothèses suivantes :

H1 Un serveur est d'autant plus important que le nombre de documents pertinents qu'il retourne est grand.

H2 Un serveur S_1 est plus pertinent qu'un serveur S_2 si les réponse de S_1 sont plus pertinentes que celles de S_2.

H3 La pertinence d'un serveur se mesure par le score du meilleur document qu'il retourne.

H4 Un serveur S_1 est plus pertinent qu'un serveur S_2, si le score moyen de ses documents pertinents est supérieur à celui de S_2 (cette hypothèse est une synthèse des deux hypothèses H1 et H2).

H5 La pertinence d'un serveur est liée au nombre de documents pertinents qu'il retourne et également au meilleur score qu'il attribue à ces derniers (les deux critères des hypothèses H1 et H3 réunis).

En se basant sur ces hypothèses, nous proposons plusieurs formulations de calcul de score capables d'attribuer un score, noté $Score(S_i)$, au serveur S_i.

1. Conformément à H1, le score d'un serveur S_i représente le nombre de documents pertinents qu'il retourne, son score $Score(S_i)$ est donc le nombre de documents de $DocChar_{nd}$ appartenant à S_i :

$$Score_1(S_i) = |\{d_{ij},\ d_{ij} \in DocChar_{nd}\}| \tag{3.3}$$

2. Concernant la deuxième hypothèse, nous proposons de définir le score d'un serveur par la somme des scores des documents pertinents qu'il a retourné. Ce score sera donc calculé comme suit :

$$Score_2(S_i) = \sum_{j | d_{ij} \in DocChar_{nd}} Sc_{ij} \tag{3.4}$$

3. En se basant sur la troisième hypothèse, nous assimilons le score d'un serveur à celui de son meilleur document pertinent. Ainsi :

$$Score_3(S_i) = \max\{Sc_{ij} | d_{ij} \in DocChar_{nd}\} \quad (3.5)$$

4. Relativement à l'hypothèse H4, le score d'un serveur se calcule comme suit :

$$Score_4(S_i) = \frac{\sum_{j|d_{ij} \in DocChar_{nd}} Sc_{ij}}{|\{d_{ij},\ d_{ij} \in DocChar_{nd}\}|} \quad (3.6)$$

5. Le score correspondant à la cinquième hypothèse se calcule comme suit :

$$Score_5(S_i) = Score_1(S_i) \times Score_3(S_i)$$
$$= |\{d_{ij} \in DocChar_{nd}\}| \times \max\{Sc_{ij} | d_{ij} \in DocChar_{nd}\} \quad (3.7)$$

3.4.4 Les approches de sélection

Nous proposons trois approches afin de définir S_q, l'ensemble des serveurs pertinents, ceux dont les résultats seront fournis à l'utilisateur. Ces approches sont dénommées :

- Classement des serveurs/sélection par nombre fixe (*CS-SNF*),
- Classement des serveurs/sélection par seuil (*CS-SS*),
- Sélection non-binaire (*SNB*).

Les deux premières débutent par une première étape classant les serveurs selon leur score (pertinence), et enchaînent ensuite avec la sélection des serveurs en tête de ce classement. Ces deux approches diffèrent dans leur façon de sélectionner les serveurs à interroger ; la première sélectionne un nombre fixe de serveurs, la deuxième, sélectionne les serveurs dont le score dépasse un seuil fixé. Dans la troisième approche, tous les serveurs sont interrogés sauf ceux qui possèdent un score nul. Cependant, on demande à chaque serveur un nombre variable de documents selon son score.

3.4.4.1 Classement des serveurs/sélection par nombre fixe (*CS-SNF*)

Après avoir classé les serveurs selon leurs scores, cette méthode sélectionne les ns les mieux classés (ns est un paramètre constant du système). Il n'est pas évident de choisir la valeur de ce paramètre. En effet, deux cas peuvent se présenter :

- si nous choisissons ns trop faible, nous courrons le risque de ne pas sélectionner tous les serveurs pertinents. Le nombre de documents pertinents dans la réponse finale diminuera (augmentation du silence), conduisant à une diminution de la précision du système ;

- si nous choisissons ns trop élevé, nous courrons le risque de sélectionner des serveurs ne contenant aucune information pertinente, et ainsi d'inclure dans la réponse finale des documents non pertinents (augmentation du bruit).

3.4.4.2 Classement des serveurs/sélection par seuil (*CS-SS*)

Dans cette deuxième approche, le classement des serveurs s'effectue en premier lieu. Ensuite, seuls les serveurs dont les scores dépassent un seuil sl, prédéfini, seront sélectionnés.
Le choix précis de la valeur de sl s'avère lié à la manière dont sont calculés les scores des serveurs. En effet, prenons l'exemple de la formule $Score_1(S_i)$, attribuant à S_i un score qui représente le nombre de document pertinents que S_i retourne en tête de sa liste de réponses. Nous pouvons choisir, alors, de considérer qu'un serveur est pertinent s'il retourne au moins un document pertinent. Dans ce cas, $sl = 1$.

3.4.4.3 Sélection non-binaire (*SNB*)

Cette approche diffère des deux précédentes dans le sens où l'approche *SNB* n'attribue pas à un serveur un jugement binaire (pertinent ou non-pertinent). *SNB* attribue implicitement un jugement quantitatif de pertinence qui se traduit par le nombre de documents que le courtier demande au serveur sélectionné. Pour calculer le nombre de documents à demander à un serveur S_i, on utilisera son score. On suppose donc que $Ndoc(S_i)$, qui est le nombre de documents que le serveur S_i doit retourner, sera proportionnel à son score. Nous avons besoin en outre de connaître le nombre de documents que l'on souhaite présenter à l'utilisateur. Ce nombre peut être donné par l'utilisateur lui même ou déterminé par le courtier. Soit Nb_{final} ce nombre, nous proposons de calculer $Ndoc(S_i)$ comme suit :

$$Ndoc(S_i) = Nb_{final} \times \frac{Score(S_i)}{\sum_{m=1}^{m=n} Score(S_m)}$$

La division par la somme des scores des serveurs sert ici à normaliser les valeurs de $Ndoc(S_i)$.

3.4.5 Récupération des listes des résultats

L'ensemble S_q des serveurs jugés pertinents étant sélectionné, les listes des résultats qu'ils retournent seront fusionnées afin de présenter à l'utilisateur la liste unique de résultats. Afin d'effectuer la fusion, les listes de résultats doivent être

disponibles au niveau du courtier. Deux cas peuvent alors se présenter selon le mode de fonctionnement des serveurs :

A. Les serveurs retournent la liste complète de leurs documents jugés pertinents. Dans ce cas il n'est pas nécessaire d'interroger une deuxième fois les serveurs sélectionnés, il suffit de fusionner les listes qu'ils ont retournées lors de leur interrogation par le courtier dans l'étape de l'acquisition des représentants de serveurs.

B. Les serveurs retournent, lorsqu'on les interroge, une partie de leur liste de réponses, c'est le cas, par exemple, des moteurs de recherche sur le Web, qui retournent leurs résultats par tranches de 10 ou 20 éléments. Dans ce cas, les serveurs sélectionnés seront ré-interrogés jusqu'à la récupération de la totalité de leur résultats qui seront ensuite fusionnés.

Concernant la fusion de ces listes, nous ne nous sommes pas concentrés, dans cette thèse, sur le domaine complexe de la fusion. Dans nos expérimentations, nous utiliserons plusieurs stratégies de fusion afin de mieux évaluer les résultats de nos approches de sélection.

3.5 Motivations de nos choix

Les trois méthodes de sélection que nous avons proposées sont inspirées de travaux existants, notre apport réside surtout dans la manière d'acquérir les représentants des serveurs et le calcul des scores de ces derniers. Dans notre méthode d'acquisition des représentants des serveurs, nous avons opté pour certains choix que nous expliquons dans ce qui suit.

1. *Pourquoi la construction des représentants des serveurs au moment de l'interrogation ?*
   Plusieurs motivations nous ont guidé vers ce choix. La première est de détecter tout éventuel changement, dans le contenu d'un serveur, qui peut concerner les documents pertinents à la requête. La seconde est de ne pas sauvegarder des informations au niveau du courtier qui nécessiteraient une mise à jour régulière d'une part, et d'autre part, engendreraient des restrictions au niveau de la nature d'informations à sauvegarder. En effet certaines informations sont coûteuses à sauvegarder à cause de leur volume, comme, par exemple, les informations de co-occurrence des termes dans les documents. Troisièmement, cette manière de procéder nous permet de prédire la performance des SRI utilisés par les serveurs. En effet, l'analyse des documents retournés par un serveur au moment de l'interrogation permet d'avoir une idée sur la capacité d'un serveur à extraire des documents pertinents à une requête donnée.

2. *Pourquoi calculer la distance seulement entre les deux premiers termes de la requête ?*
   Nous avons plusieurs arguments pour justifier notre choix de considérer seulement les deux premiers termes de la requête dans le calcul de l'indicateur de distance. Le premier est intuitif, nous pensons que les deux premiers termes de la requête (lorsque la requête est composée de plus de deux termes) sont les plus significatifs. Le second est lié aux résultats d'études récentes [WSJS01] et [SWJT01] qui confirment que, approximativement, 60% des utilisateurs expriment leur besoin d'information par des requêtes composées de un ou de deux termes. Enfin, notre désir de minimiser le temps d'analyse des documents.

3. *Pourquoi veut-on une méthode plus efficace dans la détermination de la pertinence des d_{ij} ?*
   Nous pensons que si nous déterminons mieux la pertinence d'un document, ce jugement permet de prédire la pertinence du serveur d'où le document provient. Nous présentons dans le chapitre 4 les expérimentations qui appuient cette position.

4. *Pourquoi a-t-on utilisé distance et co-occurrence ?*
   Nous pensons que ces deux métriques permettent une meilleure qualité de recherche. Leur utilisation n'est pas répandue en raison de leur coût. En effet, il est extrêmement coûteux de récolter et sauvegarder les données en question (co-occurrence et proximité) pour des millions de documents et des milliers de termes et de combinaisons de termes.

3.6 Apports de nos approches

3.6.1 Sélection de serveurs vs. sélection de collections

La première différence de notre approche comparée à l'ensemble des approches connues dans le domaine de la RID (à part [CBH00]) réside dans le fait que nous cherchons à déterminer les serveurs, et non pas les collections, les plus pertinents. En effet, détecter une collection qui contient des documents pertinents n'a aucun sens si le moteur de recherche qui opère sur cette collection n'est pas en mesure de les retrouver. Dans cette optique, nous avons décidé d'analyser les premiers documents qu'un serveur retourne pour une requête donnée afin de déterminer sa performance. Cette stratégie permet de vérifier deux faits à la fois. Primo, s'il existe des documents pertinents dans la collection (dans ce cas, ils devraient être les premiers à être retournés). Secundo, si le moteur de recherche est capable d'extraire ces documents.

3.6.2 Absence de coopération vs. coopération nécessaire

Notre approche ne requiert aucune forme de coopération entre serveurs et courtier. En effet, l'interrogation des serveurs et le chargement des textes des premiers documents retournés sont effectués par le courtier et sont perçues par les serveurs comme n'importe quelle action générée par un utilisateur. La plupart des autres méthodes de sélection connues (voir table 2.2) ont besoin d'un minimum de coopération entre serveurs et courtier. Ce minimum consiste dans l'accord des serveurs à être interrogés par le courtier de manière répétitive.

3.6.3 Courtier transportable vs. courtier centralisé

Notre courtier peut être facilement installé sur la machine de l'utilisateur. En effet, vu que notre approche ne nécessite pas la sauvegarde de données, l'algorithme de sélection est facilement transportable. Alors que la plupart des méthodes étudiées ont besoin de sauvegarder et donc de mettre à jour des données centrales ce qui rend leur transportablité difficile. Cette caractéristique de notre approche permet de ne nécessiter aucune forme de coopération (passive ou active) entre courtier et serveurs. Effectivement, si chaque usager installe notre courtier sur sa propre machine, les serveurs se comporteront exactement comme si l'usager utilisait leur propre interface de recherche, et non pas comme si un courtier les interrogeait. Les interrogations dans ce dernier cas étant récurrentes et proviennent de la même source (la machine où se trouve le courtier). En outre, le courtier se trouvant sur la machine de l'utilisateur, il n'y a pas de risque d'encombrement lié à l'utilisation simultanée du courtier par de nombreux utilisateurs.

3.6.4 Documents vs. références aux documents

Notre approche analyse le contenu de certains documents pour déterminer leur pertinence pour la requête, et cette pertinence détermine l'utilité des serveurs. Parmi les méthodes que nous avons étudiées, aucune d'elles n'effectue l'analyse du contenu des documents au moment de l'interrogation. Certes, il existe quelques méthodes qui ont recours à l'analyse du contenu des documents pour effectuer la sélection comme, par exemple, la méthode proposée par Craswell [Cra00]. Dans ce dernier cas, la différence avec notre approche réside dans le fait que cette analyse n'est pas effectuée au moment de l'interrogation. Dans ce même type d'approche, les travaux de Callan et al. [CCD99] visent à analyser les documents afin de construire les représentants des serveurs. D'autre méthodes encore effectuent l'analyse des documents chargés afin d'effectuer la fusion comme, par exemple, Selberg et al. [SE95] qui chargent les documents afin de vérifier que leur contenu est encore adapté à la requête et qu'il n y a pas eu des changements entre l'indexation et le contenu actuel. Lawrence et Giles [LG98] analysent aussi les documents chargés afin de les classer selon leur pertinence.

Nous sommes conscients que l'analyse des documents au moment de l'interrogation peut être coûteuse. Néanmoins, nous pouvons nous permettre d'utiliser des fonctions d'appariement plus performante, comme par exemple celle utilisant la proximité des mots clés dans les documents. Ces fonctions, qui en temps normal sont coûteuses deviennent tout à fait raisonnables dans notre cas, car nous analysons un nombre restreint de documents.
Dans la section 4.4.8.3 du chapitre 4 nous présentons des expériences afin de vérifier si l'utilisation des scores locaux est efficace. Si cela s'avère intéressant, nous n'aurions pas besoin de charger les documents.

3.6.5 Construction des représentants des serveurs à la demande vs. à l'avance

Notre méthode acquiert les informations qui permettent d'effectuer la sélection de serveurs au moment de la réception de la requête. La seule méthode qui adopte cette stratégie est celle de Hawking et al. [HT99]. La différence avec notre approche réside dans le fait que les informations sont extraites par l'analyse des textes des documents dans notre modèle tandis que ces informations sont retournées par les serveurs eux-mêmes grâce à leur coopération dans le modèle de Hawking.

3.7 Conclusion

Notre but principal dans cette thèse était de concevoir une méthode de sélection qui soit, d'une part, la moins contraignante possible et, d'autre part, qui atteigne une performance raisonnable. Dans ce chapitre, nous avons présenté notre approche satisfaisant la première partie de notre but à savoir : concevoir une méthode qui ne nécessite ni coopération entre serveurs, ni mise à jour au niveau du courtier et qui, en plus, est extensible. En effet, notre algorithme de sélection peut être installé au niveau du client (l'utilisateur) permettant ainsi d'éviter un accord des serveurs à être interrogés par un courtier, ce que l'on a appelé la coopération passive. D'un autre côté, comme les représentants des serveurs sont spécifiques à chaque requête, leur construction se fait au moment de la réception de la requête et se base sur l'analyse des textes des documents retournés par les serveurs. Ainsi, aucune coopération n'est exigée. Enfin, la mise à jour des données n'est pas nécessaire, vu que ces données sont spécifiques aux requêtes et qu'elles ne sont, de ce fait, pas sauvegardées.

Nous proposons, dans le prochain chapitre, les différentes expérimentations que nous avons effectuées afin de vérifier la performance de nos approches de sélection.

Chapitre 4

Évaluation

Table des matières

4.1 Introduction

Ce chapitre a pour objet l'évaluation de nos approches de sélection. Basé sur une forte tradition empirique, nous utilisons des collections-tests afin de vérifier la performance de nos approches et aussi afin de les comparer à d'autres approches telles que l'approche centralisée ou celle proposée par *CORI*.

Nous commençons ce chapitre par la description des deux collections tests que nous avons utilisées pour effectuer nos expérimentations. Dans la section 4.3, nous décrivons les environnements de test ainsi que les méthodes de sélection de serveurs, les méthodes de fusion des résultats et les méthodes d'évaluation employées. Nous détaillons ensuite, dans la section 4.4, les expérimentations réalisées, leurs résultats et l'analyse de ces dernières. Nous présentons, dans la section 4.5, une étude comparative d'approches présentées dans ce chapitre. Nous dressons enfin dans la conclusion un bilan sur les résultats obtenus.

4.2 Les collections-tests

Une collection-test en RI est constituée de :

- un ensemble de documents ;
- un ensemble de requêtes (requêtes-tests) ;
- un ensemble de jugements de pertinence.

Dans la plupart des collections-tests disponibles, les documents sont des unités textuelles. Une requête est une brève description du besoin d'information de l'utilisateur. Pour chaque requête, nous disposons d'un jugement de pertinence qui identifie l'ensemble des documents pertinents c'est-à-dire les documents qui satisfont le besoin d'information de l'utilisateur.

Pour nos expérimentations nous avons choisi d'utiliser deux des collections de TREC à savoir TREC8 et TREC9. TREC (Text REtrieval Conference) est une série de campagnes d'évaluation annuelles sponsorisée par le NIST (National Institute of Standards and Technology) et la DARPA (Defense Advanced Research Projects Agency). Chaque année, un corpus de documents et un ensemble de requêtes sont proposés aux équipes participantes. Les résultats retournés par les systèmes de ces équipes sont jugés par des opérateurs humains décidant si tel ou tel document est pertinent. Cette opération, la plus coûteuse, forme les jugements de pertinence. Nous présentons un exemple de document (respectivement de requête et de jugement de pertinence) de TREC dans l'annexe A (respectivement B et C). Les collections de TREC sont prisées, car la majorité des évaluations dans le domaine de la RI sont basées sur de telles collections et, d'autre part, TREC propose des volumes assez importants de données textuelles. Nous détaillons dans ce qui suit les deux collections tests utilisées pour nos expérimentations.

4.2.1 TREC8

La collection TREC8 [HVCB99a] correspond au corpus de la huitième conférence de TREC. Elle contient 528 155 documents dont le volume est de 1 904 méga-octets. Ces documents sont extraits des quatre sources suivantes :

- les articles du journal *Finacial Times* (FT) ;
- les articles du journal *Federal Register* (FR) ;
- les articles du journal *Foreign Broadcast Information Service* (FBIS) ;
- les articles du journal *Los Angeles Times* (LA).

Afin d'exploiter ce corpus, nous disposons de cinquante requêtes (numérotées par TREC : 401 à 450) et leurs jugements de pertinence. Ces requêtes couvrent un large éventail de thèmes, tels que « *foreign minorities, Germany* », « *U.S., investment, Africa* », « *antibiotics ineffectiveness* ». Une requête comporte en moyenne 2 termes avec un écart type de 0, 97.

4.2.2 TREC9

La collection TREC9 (Web Track, 10 giga-octets) [VH00] est le corpus Web Track de la neuvième conférence de TREC. Cette collection est constituée de 1 692 096 pages Web écrites en anglais. La taille de ce corpus est de 11 033 méga-octets. Le volume de TREC9 est approximativement six fois plus grand que celui de TREC8.
Nous avons également cinquante requêtes (numérotées par TREC : 451 à 500). Ces requêtes sont recueillies parmi celles soumises au moteur eXcite[1]. Les requêtes concernent des domaines très variés. Des exemples de requêtes sont : « *nirvana* »,« *do beavers live in salt water* ». La taille d'une requête est en moyenne de 2, 4 termes (après la suppression des mots vides) avec un écart type de 0, 65.

Les requêtes de TREC sont composées de quatre champs (voir annexe C page 121 pour un exemple) :

- "num" qui représente le numéro de la requête ;
- "title" qui représente la requête telle qu'elle a été soumise par un utilisateur ;
- "description" est une description du besoin d'information de l'utilisateur ;
- "narrative" est une description des caractéristiques que les documents pertinents doivent avoir.

1. http://www.excite.com/

4.3 Environnements et méthodes utilisés

Pour mener nos expérimentations, il a fallu simuler un environnement distribué, décider quels seraient les différents systèmes à tester, choisir une méthode de fusion à appliquer sur les listes retournées par les serveurs sélectionnés et enfin choisir la méthode d'évaluation de ces systèmes.

4.3.1 Construction des environnements de test

Afin d'effectuer nos évaluations, nous avons créé un environnement similaire à celui auquel nos approches sont dédiées. Pour cela, nous avons d'abord divisé chacun des deux ensembles de documents de nos collections-tests en sous-ensembles, que nous appelons ici collection. Nous avons ensuite attribué à chaque collection un modèle de recherche d'information (MRI). Nous définissons pour la suite un environnement comme étant un ensemble de serveurs, un serveur étant une collection associée à un MRI. Un environnement est donc représenté comme suit :

Environnement := $[(Collection_1, MRI_1),...,(Collection_x, MRI_x)]$

4.3.1.1 Subdivision en collections

L'ensemble des documents de la collection-test TREC8 est découpée pour cette étude selon l'origine du document et ainsi quatre collections sont obtenues : FT, FR, FBIS et LA que nous nommons par la suite, respectivement TREC8.1 à TREC8.4.
L'ensemble des documents de la collection-test TREC9 a été découpé en huit collections disjointes possédant approximativement le même nombre de documents. Nous nommons ces collections : TREC9.1 à TREC9.8.
Différentes statistiques concernant les collections générées par les deux subdivisions des collections TREC8 et TREC9 sont présentées respectivement sur les tableaux 4.1 et 4.2.
La deuxième colonne *Taille(MO)*, représente la taille en méga-octet d'une collection, *Nb. Doc.* le nombre de documents qu'elle contient, *Nb. Req. Pert.* le nombre de requêtes pour lesquelles la collection contient au moins un document pertinent.

TAB. 4.1 – *Statistiques sur l'ensemble de documents de la collection* TREC8 *et les collections générées.*

Collection	Taille(MB)	Nb. Doc.	Nb. Req. Pert.
TREC8.1 (FT)	564	210 158	49
TREC8.2 (FR)	395	55 630	18
TREC8.3 (FBIS)	470	130 471	43
TREC8.4 (LA)	475	131 896	45
TREC8	1 904	528 155	50

TAB. 4.2 – *Statistiques sur l'ensemble de documents de la collection* TREC9 *et les collections générées.*

Collection	Taille(MB)	Nb. Doc.	Nb. Req. Pert.
TREC9.1	1 325	207 485	28
TREC9.2	1 474	207 429	44
TREC9.3	1 438	221 916	38
TREC9.4	1 316	202 049	21
TREC9.5	1 309	203 073	22
TREC9.6	1 311	215 451	24
TREC9.7	1 336	200 146	25
TREC9.8	1 524	234 547	43
TREC9	11 033	1 692 096	50

4.3.1.2 Simuler un SRID

Les SRIDs que nous construisons en vue d'évaluer nos approches de sélection de serveurs sont constitués d'un courtier et d'un ensemble de serveurs cohabitant sur la même machine. Il n'y a donc pas une distribution physique sur différentes machines.
Chaque serveur se compose d'une collection (l'une des collections résultant de l'étape de subdivision présentée dans la section précédente) et d'un MRI que nous lui attribuons. Nous décrivons ici les MRI utilisés dans nos expérimentations.
Pour décrire un modèle de recherche d'information, il convient de définir la représentation des documents, la représentation des requêtes et la façon dont l'appariement entre un document et une requête se réalise. Dans notre étude nous avons choisi très classiquement de représenter les documents et les requêtes comme des vecteurs de termes pondérés. L'appariement est le produit scalaire entre les deux vecteurs représentant respectivement le document et la requête, selon la proposition de Salton [Sal89]. Dans la pratique, nous utilisons le système Smart [Sal89], adapté par J. Savoy et Y. Rasolofo, afin de réaliser l'indexation et l'appariement.

Dans ce qui suit, pour faire référence à un MRI, nous adoptons la notation utilisée par Smart. Smart représente un MRI à l'aide d'un couple (fp_{td}, fp_{tr}), où, fp_{td} est la fonction de pondération d'un terme t dans un document d et fp_{tr} est la fonction de pondération d'un terme t dans la requête. Une fonction de pondération est représentée par trois lettres codant chacune une fonction. La première lettre code une fonction de la fréquence de t dans d (ou dans q). La deuxième code une fonction de la fréquence de t dans le corpus. Enfin, la troisième lettre code une fonction de normalisation. Nous présentons dans l'annexe A les différents codes manipulés par Smart. Ce système permet de réaliser différentes méthodes d'indexation (différentes pondérations) ; entre autres les méthodes que nous avons utilisées dans nos expérimentations, à savoir :

- le modèle (nnn , nnn) où seule la fréquence d'occurrence d'un terme dans le document (ou dans la requête) est prise en compte dans l'appariement ;
- le modèle (Okapi, npn), le modèle probabiliste proposé dans [RW95] dont la pondération prend en compte la fréquence d'occurrence d'un terme, l'inverse de la fréquence document d'un terme et la longueur du document. Ainsi, le poids $p(t,d)$ de chaque terme t apparaissant dans un document d est calculé comme suit :

$$p(t,d) = (k_1 + 1)\frac{tf_{td}}{K + tf_{td}} \tag{4.1}$$

tel que $K = k \cdot \left[(1-b) + b\frac{l_d}{moy\_l}\right]$

où : l_d la longueur du document d,
moy_l la moyenne des longueurs des documents,
b une constante positionnée à 0.9,
k une constante positionnée à 2,
k_1 une constante positionnée à 1.2,
tf_{td} la fréquence d'occurrence du terme t dans le document d.

Le poids $p(t,q)$ d'un terme t dans la requête q est calculé comme suit :

$$p(t,q) = \frac{tf_{tq}}{k_3 + tf_{tq}} \cdot \log\left[\frac{nb - df_t}{df_t}\right] \tag{4.2}$$

où : tf_{tq} la fréquence d'occurrence du terme t dans q,
df_t le nombre de document contenant le terme t,
nb le nombre de document du corpus recherché,
k_3 une constante positionnée à 1000.

4.3.2 Description des approches concernées par nos expérimentations

Afin de pouvoir juger de la performance de nos approches, il ne suffit pas de calculer cette performance de manière absolue, mais il faut aussi pouvoir l'éva-

luer de manière relative. Nous avons donc choisi de comparer la performance de nos approches à :

a) celle de l'approche centralisée, notre but étant d'atteindre cette performance ;

b) celle de l'approche qui interroge tous les serveurs sans sélection préalable. Notre but par cette comparaison est de vérifier s'il n'est pas vain d'effectuer nos approches de sélection ;

c) celle de *CORI*, car c'est l'approche la plus utilisée à des fins de comparaison dans le domaine de la RID.

4.3.2.1 L'approche centralisée (*Centr*)

L'approche centralisée fait partie de notre référentiel car le but de la RID est d'atteindre, voire de dépasser, la performance des SRIC tout en offrant des solutions aux problèmes inhérents à la centralisation. Cependant, la plupart des études expérimentales montrent que la performance des SRIDs proposés s'avère inférieure à celle des SRICs correspondants[2]. Parmi ces études nous citons [SR00, VF95, CBH00, XC98, XC99]. Néanmoins, d'autres études [PFC00, XC99] ont montré que si un SRID effectuait une bonne sélection et une bonne fusion alors ses résultats peuvent être meilleurs que ceux du SRIC correspondant. Une bonne sélection est celle qui permet d'optimiser le nombre de documents pertinents à retourner à l'utilisateur. Une bonne fusion est celle qui classe les documents pertinents en tête de la liste retournée à l'utilisateur.

4.3.2.2 L'approche triviale : pas de sélection (*PasSel*)

Cette approche est la plus courante parmi les méta-moteurs sur le Web et son principe est simple. À la réception d'une requête, le courtier la diffuse à tous les serveurs sans exception.

4.3.2.3 *CORI*

Nous choisissons d'inclure *CORI* dans notre étude comparative à cause de sa notoriété. En effet, elle a été l'objet de comparaison dans plusieurs études, notamment : [YL97, FPC+99, CBH00, LC00, PFC00, SR00].

La méthode consiste à considérer chaque collection comme un unique document. Ce document, gigantesque, est la concaténation de tous les documents de la collection en question. La procédure de classement des serveurs, pour une requête

2. Lorsque nous disons "un SRIC correspondant à un SRID" nous parlons du SRIC dont la base de documents est l'union des corpus de tous les serveurs du SRID

donnée, est assimilé au classement d'un ensemble de documents dans un SRI traditionnel et cette procédure peut donc être réalisée par un algorithme classique de RI. Afin d'effectuer ce classement, des informations concernant les serveurs sont utilisées, elles sont sauvegardées au niveau du courtier. Ces informations sont les suivantes : DF_{ij} (Document Frequency) et ICF_j(Inverse Collection Frequency).
Où :

- DF_{ij} est la fréquence document du terme t_j ;
- ICF_j est calculé comme suit :

$$ICF_j = \frac{\log(\frac{|S|+0.5}{CF_j})}{\log(|S|+1.0)}$$

où CF_j est la fréquence collection du terme t_j.

À partir de ces informations, et pour une requête donnée, un score s_i est attribué à chaque serveur S_i du système.

$$s_i = \frac{1}{m} \cdot \sum_{j=1}^{m} s(t_j|S_i)$$

$$s(t_j|S_i) = \alpha + (1-\alpha) \cdot (\frac{DF_{ij}}{DF_{ij}+K}) \cdot ICF_j$$

avec

$$K = k \cdot ((1-b) + b \cdot \frac{l_i}{\bar{l}_i})$$

où : m est le nombre des termes de la requête ;
l_i est le nombre de termes que contient S_i ;
$\bar{l}_i$ est la moyenne des l_i ;
α, k et b sont des constantes.

On remarquera que la formule précédente est l'une des variations de l'approche $tf \times idf$ en prenant le soin de remplacer tf par DF et idf par ICF.

Dans nos expérimentations, les paramètres précédents sont positionnés aux valeurs suivantes comme suggéré par Callan & al. [CLC95], $\alpha = 0,4$; $k = 200$; $b = 0,75$.

4.3.2.4 Nos trois approches : *CS-SNF*, *CS-SS*, *SNB*

Dans nos expérimentations, nous testons évidemment nos trois approches *CS-SNF*, *CS-SS* et *SNB* présentées en détails dans le chapitre 3. Nous varions les valeurs des différents paramètres utilisés par ces approches afin de déterminer leurs valeurs optimales. Nos approches utilisant ces valeurs optimales seront comparées aux autres approches.

4.3.3 Méthodes de fusion utilisées

L'étape de fusion des résultats est une étape nécessaire afin de pouvoir évaluer les résultats retournés à l'utilisateur. Nous utilisons dans nos expérimentations les méthodes de fusion suivantes :

- *RSM* (Raw Score Merging ou fusion par le score) (voir 2.5). Pour rappel, cette méthode fusionne les listes de résultats selon les scores locaux des documents, c'est à dire, les scores attribués par les serveurs retournant ces documents. Cette méthode suppose que les scores locaux provenant de différents serveurs sont disponibles et comparables.
- *CoriF* la méthode de fusion proposée par Callan et al. [CLC95]. Après l'étape de sélection de serveurs où *CORI* attribue un score s_i à chaque serveur S_i de l'ensemble S des serveurs connus par le système, s_i est utilisé afin de pondérer les scores des documents retournés par le serveur S_i. Le poids w_i associé au serveur S_i est calculé comme suit :

  $$w_i = 1 + |S| * (s_i - \bar{s})\bar{s}$$

  où $\bar{s}$ est la moyenne des scores des serveurs et |E| le cardinal de l'ensemble E.

  Le score d'un document retourné par le serveur S_i est multiplié par w_i. Les documents provenant de tous les serveurs sélectionnés sont alors classés selon ce nouveau score.
- *LMS* (Length to calculate Merging Score : utilisation de la longueur des réponses[3] pour calculer le score servant à la fusion).
  Cette approche calcule dans un premier temps un score s_i pour chaque serveur S_i ayant été interrogé. L'idée est de pondérer les scores locaux des documents à fusionner afin de favoriser les documents retournés par les serveurs ayant un score supérieur au score moyen de tous les serveurs et de pénaliser les autres documents.
  Le score d'un serveur est calculé sur la base de la longueur de la liste qu'il retourne. Le calcul de s_i est basé sur l'intuition que plus une liste de réponse est longue plus la chance qu'elle contienne des documents pertinents augmente.

3. nombre de documents retournés par un serveur donné

Le score s_i d'un serveur S_i, pris dans l'ensemble des serveur S connus par le système, est calculé comme suit :

$$s_i = \log(1 + \frac{l_i \cdot K}{\sum_{j=1}^{|S|} l_j})$$

où K est une constante (K=600 dans les expérimentations) et l_i est le nombre de documents retournés par le serveur S_i.

Basé sur ce score, un poids w_i est calculé pour chaque serveur S_i comme suit :

$$w_i = 1 + \frac{(si - \bar{s})}{\bar{s}}$$

où $\bar{s}$ est la moyenne des scores des serveurs.
Les documents sont alors classés selon le nouveau score qui est le produit du score local et du poids calculé ci-dessus.

4.3.4 Mesures de performance utilisées

Nous avons exposé dans la section 2.3 du chapitre 2 les différentes méthodes utilisées afin d'évaluer un SRID à savoir *EvalPerf* et *EvalSel*. Pour rappel, *EvalPerf* évalue la performance du système par rapport aux résultats qu'il retourne. Cette évaluation est associée à la satisfaction qu'un utilisateur peut avoir du système. La méthode *EvalSel* évalue seulement la sélection. Ces deux méthodes d'évaluations seront utilisées pour plus de clarté concernant la performance de nos approches.

Pour effectuer l'évaluation *EvalPerf*, nous utilisons le programme *trec_eval* afin de calculer la précision moyenne (PM) à onze points fixes de rappel. Nous considérons la règle d'usage qui stipule qu'une différence de précision moyenne entre deux systèmes n'est significative que si elle dépasse 5%, pour une différence de moins de 5% on considère que les deux systèmes apportent la même performance [JB77].

Concernant *EvalSel*, nous avons opté pour l'utilisation des deux mesures P_p et R_p décrites ci-dessous.

- P_p détermine la proportion des serveurs pertinents parmi les p serveurs sélectionnés. Un serveur est pertinent lorsqu'il contient au moins un document pertinent. P_p est calculé comme suit :

$$P_p = \frac{|NbServPert_p|}{p}$$

où : $|NbServPert_p|$ est le nombre de serveurs pertinents parmi les p serveurs sélectionnés.
p est le nombre des serveurs sélectionnés.

- R_p détermine la proportion des documents pertinents appartenant aux serveurs sélectionnés lorsque p serveurs sont sélectionnés. R_p est calculé alors comme suit :

$$R_p = \frac{\sum_{i=1}^{p} r_i}{Rel}$$

où : r_i est le nombre des documents pertinents contenus dans le serveur de rang i.
Rel est le nombre des documents pertinents contenus dans l'ensemble des serveurs du système.
p est le nombre des serveurs sélectionnés.

Le choix de ces mesures se justifie par notre souhait de vérifier d'une part si une approche de classement de serveurs classent les serveurs pertinents en tête du classement et les serveurs non-pertinents en bas du classement. Ceci est fait à l'aide de la métrique P_p. Et d'autre part, si les serveurs pertinents (i.e. qui contiennent au moins un document pertinent) sont classés dans l'ordre de leur pertinence. La pertinence d'un serveur étant ici le nombre de documents pertinents contenu dans le serveur. Cette dernière vérification est effectuée à l'aide de la métrique R_p.

4.4 Expérimentations, résultats et discussions

Nous présentons dans cette section notre étude expérimentale. Pour chacune des expérimentations réalisée nous présenterons sa description, son but, les résultats obtenus et enfin une discussion de ces derniers.

Dans ce qui suit, le couple (Sel, Fus) dénote un SRID qui utilise une méthode Sel pour sélectionner les serveurs et une méthode Fus pour fusionner les résultats retournés par les serveurs sélectionnés.

4.4.1 L'approche centralisée (*Centr*)

4.4.1.1 But

Le but de cette expérimentation est de déterminer la performance de l'approche *Centr* dans les deux collections test lorsque différents MRI sont utilisés. Cette expérience nous permet de choisir le MRI le plus performant pour le reste des expérimentations.

4.4.1.2 Description

Chaque corpus test est donc indexé en entier sans la subdivision en collections. L'indexation est effectuée à l'aide du programme Smart. Ce dernier offre la possibilité d'appliquer de nombreux MRI.
Le processus suivant décrit le déroulement de cette expérimentation.

*Pour chaque corpus (*TREC8 *et* TREC9*)* :
- *Pour chaque MRI parmi ceux applicables par Smart* :
  - Indexer la totalité des documents du corpus à l'aide du MRI
  - *Pour chaque requête-test q* :
    - Interroger l'index par le biais de Smart ;
    - récupérer les résultats retournés ;
    - appliquer *trec_eval* afin de calculer PM_q la précision moyenne des résultats.
- Calculer PM la moyenne des PM_q

4.4.1.3 Résultats

Dans les deux tables 4.3 et 4.4 (correspondant respectivement aux résultats avec les deux corpus TREC8 et TREC9) nous présentons les résultats de cette expérimentation. La première colonne de chaque table contient les MRI testés, et la deuxième colonne les PM obtenues. Enfin, la troisième colonne montre la perte en précision moyenne par rapport au modèle Okapi qui est, significativement, le plus performant des modèles.

TAB. 4.3 – *Approche Centr : résultats retournés par différents MRI appliqués à* TREC8

Modèle de RI	PM	Diff. % Okapi
(Okapi , npn)	0.2566	-
(dnu , dtn)	0.2403	-6.4%
(atn , ntc)	0.2275	-19.1%
(Lnu , ltc)	0.2163	-15.7%
(lnc , ltc)	0.1444	-43.7%
(ltc , ltc)	0.1360	-47.0%
(ntc , ntc)	0.1221	-52.4%
(bnn , bnn)	0.1151	-5.13%
(nnn , nnn)	0.0447	-82.6%

TAB. 4.4 – *Approche Centr : résultats retournés par différents MRI appliqués à* TREC9

Modèle de RI	*PM*	Diff. % Okapi
(Okapi , npn)	0.1960	-
(dnu , dtn)	0.1542	-21.32%
(atn , ntc)	0.1459	-25.56%
(Lnu , ltc)	0.1681	-14.23%
(lnc , ltc)	0.0379	-80.66%
(ltc , ltc)	0.0520	-73.46%
(ntc , ntc)	0.0800	-59.18%
(bnn , bnn)	0.0513	-73.82%
(nnn , nnn)	0.0406	-79.28%

4.4.1.4 Discussion

Les résultats obtenus montrent que le modèle probabiliste Okapi est plus performant que le reste des modèles. Le modèle booléen (modèle (bnn , bnn)), s'avère plus performant que le modèle (nnn , nnn) utilisant la fréquence des termes. Une autre constatation concerne le modèle vectoriel classique (modèle (ntc, ntc) ou tf*idf) qui occupe une position médiocre dans le classement, à savoir la septième position dans TREC8, et la cinquième dans TREC9.

4.4.2 L'approche triviale (*PasSel*)

4.4.2.1 But

Cette expérimentation a pour objectif de mesurer la performance d'un SRID qui n'effectue pas de sélection de serveurs et envoie chaque requête à tous les serveurs sans exception. Les résultats de cette expérience nous permettront de démontrer, par la suite, l'utilité d'une phase de sélection de serveurs.

4.4.2.2 Description

Cette expérimentation suit les étapes du processus suivant.

Pour chaque environnement
*[(*TREC8*.1 , Okapi),..,(*TREC8*.4 , Okapi)] et*
*[(*TREC9*.1 , Okapi),..,(*TREC9*.8 , Okapi)] :*
{ *Pour chaque requête-test q :*
{
- soumettre la requête à tous les serveurs ;
- récupérer les réponses provenant des serveurs ;

Pour tout algorithme **Fus** *parmi les algorithmes de fusion RSM, LMS, CoriF :*
{
- appliquer l'algorithme de fusion *Fus* sur les résultats ;
- appliquer *trec_eval* afin de calculer PM_q la précision moyenne des résultats.

- Calculer la moyenne PM des PM_q

4.4.2.3 Résultats

La table 4.5 contient les précisions moyennes que l'on a obtenues lorsque tous les serveurs sont interrogés. Les valeurs entre parenthèses indiquent la différence de performance par rapport à l'approche centralisée.

TAB. 4.5 – *Approche PasSel: PM obtenue en combinant l'approche PasSel à différents algorithmes de fusion, dans deux environnements différents.*

	Environnement	
Système	[(TREC8.1 , Okapi), : (TREC8.4 , Okapi)]	[(TREC9.1 , Okapi), : (TREC9.8 , Okapi)]
(*PasSel*, *RSM*)	0.2397 (-6.59%)	0.1832 (-6.53%)
(*PasSel*, *CoriF*)	0.2416 (-5.85%)	0.1847 (-5.76%)
(*PasSel*, *LMS*)	0.2462 (-4.05%)	0.1932 (-1.42%)

4.4.2.4 Discussion

Dans la table 4.5 nous avons les résultats de la fusion des mêmes listes[4] par différentes méthodes de fusion. En plus de nous permettre de connaître la performance de l'approche *PasSel*, les résultats de cette expérimentation nous permettent de comparer les méthodes de fusion en question.

4. les listes provenant de tous les serveurs

La table 4.5 nous permet de faire les deux constatations suivantes.

1. Les performances de l'approche *PasSel* apparaissent être en dessous de celles de l'approche centralisée comme on pouvait s'y attendre. En effet, sélectionner tous les serveurs pour n'importe quelle requête signifie augmenter le risque d'inclure les réponses des serveurs qui ne contiennent aucun document pertinent et de ce fait augmenter le bruit dans la réponse finale. Et l'augmentation du bruit dans une liste de réponses diminue la précision moyenne de cette liste.
   Nous remarquons néanmoins, que la différence de performance entre le système utilisant *LMS* et l'approche *Centr* n'est pas significative. Les résultats d'un SRID dépendent donc aussi de la méthode de fusion utilisée et pas seulement de la méthode de sélection.

2. Concernant les méthodes de fusion, *LMS* parait la plus performante des trois approches dont il est question dans cette table. En effet, l'amélioration de performance que *LMS* présente, par rapport à *RSM* et *CoriF*, est significative. De plus, *LMS* permet d'atteindre la performance de l'approche centralisée puisque la différence de performance que le système (*PasSel*, *LMS*) présente par rapport à *Centr* n'est pas signifiicative.

   La méthode *RSM* parait la moins performante des trois méthodes avec une dégradation de performance de l'ordre de -7% en moyenne par rapport à l'approche *Centr*.

4.4.3 L'approche centralisée (*Centr*) vs. l'approche distribuée

4.4.3.1 But

Les travaux de Powell [PFC00] démontrent qu'un SRID effectuant une *bonne* sélection atteint une performance meilleure qu'un SRIC. Nous avons tenu dans nos travaux à vérifier si, dans nos environnements de test, un SRID pouvait atteindre la performance d'un SRIC pour le même ensemble de documents indexés.
La question à laquelle nous nous sommes confronté est de savoir ce qu'est une *bonne* méthode de sélection? Powell et al. [PFC00] ont suggéré cette définition : c'est la sélection des serveurs conteneant le plus grand nombre de documents pertinents. Nous proposons une autre définition : une *bonne* méthode de sélection est celle qui sélectionne tout serveur contenant au moins un document pertinent. Nous notons cette sélection *Optimal1*.
Le but est donc de vérifier qu'il n'est pas vain de chercher à proposer une méthode de sélection qui permette à un SRID de surpasser la performance d'un SRIC.

4.4.3.2 Description

Comme nous disposons des jugements de pertinence pour les deux collections-tests, nous pouvons déterminer la sélection optimale de serveurs pour chacune des requêtes-tests. Le processus suivant décrit les étapes de cette expérimentation :

Pour chaque environnement
*[(*TREC8*.1 , Okapi),..,(*TREC8*.4 , Okapi)] et*
*[(*TREC9*.1 , Okapi),..,(*TREC9*.8 , Okapi)]* :
- *Pour chaque requête-test q* :
  - - déterminer la sélection optimale pour la requête ;
    - interroger à l'aide de Smart les serveurs sélectionnés ;
    - *Pour tout algorithme* **Fus** *parmi les algorithmes de fusion RSM, LMS, CoriF* :
      - - fusionner les résultats par **Fus** ;
        - appliquer *trec_eval* afin de calculer PM_q la précision moyenne des résultats.
- Calculer la moyenne PM des PM_q

4.4.3.3 Résultats

Nous rapportons dans la table 4.6 les résultats de l'expérimentation décrite ci dessus. Les valeurs entre parenthèses indiquent la différence de précision moyenne par rapport à l'approche centralisée *Centr*.
TAB. 4.6 – *Approche Optimal1 : PM obtenus lors de la combinaison de l'approche Optimal1 avec différentes approches de fusion.*

	Environnement	
Système	[(TREC8.1 , Okapi), : (TREC8.4 , Okapi)]	[(TREC9.1 , Okapi), : (TREC9.8 , Okapi)]
(*Optimal1*, *RSM*)	0.2543 (-0.90%)	0.2097 (+6.98%)
(*Optimal1*, *CoriF*)	0.2533 (-1.29%)	0.2142 (+9.28%)
(*Optimal1*, *LMS*)	0.2480 (-3.35%)	0.2144 (+9.38%)

4.4.3.4 Discussion

Les résultats qu'on obtient en appliquant la sélection *Optimal1* (supposée une *bonne* sélection) sont différents selon le corpus utilisé. En effet, sur TREC8,

comparés aux résultats de l'approche centralisée, les systèmes utilisant *Optimal1* n'entraînent aucune amélioration de performance, et ce, quelque soit la méthode de fusion utilisée. Néanmoins, la différence de performance n'est pas significative (inférieure à 5%) les approches comparées sont donc jugées équivalentes.
Sur le corpus TREC9, les résultats sont plus encourageants car, quelle que soit la méthode de fusion utilisée, nous obtenons une amélioration significative par rapport à l'approche centralisée.

Cette expérimentation, d'une part, confirme qu'il existe (au moins) une méthode de sélection (une *bonne* sélection) qui permettent d'obtenir une performance similaire ou supérieure aux résultats obtenus par l'approche centralisée. D'autre part, cette expérimentation confirme que sélectionner les collections qui contiennent au moins un document pertinent s'avère être une *bonne* sélection.

4.4.4 L'approche *PasSel* vs. l'approche *Optimal1*

Dans cette section nous comparons les performances d'une approche sans sélection (voir 4.4.2) avec celles obtenues dans le cadre d'une sélection optimale (voir 4.4.3). Nous avons rassemblé dans la table 4.7 les résultats présentées dans les tables 4.5 et 4.6 afin de mieux les comparer. Les colonnes *diff.* indiquent la différence de performance entre les approches *PasSel* et *Optimal1*.
Notre but est de vérifier que la sélection optimale apporte une amélioration de performance, du moins dans nos corpus, comparée à une approche où la totalité des serveurs est sélectionnée. Théoriquement, l'opération de sélection de serveurs devrait augmenter l'efficacité d'un système. En effet, d'un côté, la sélection permet d'éliminer les serveurs qui ne contiennent pas de documents pertinents, et par conséquent, la liste finale des résultats comprendra moins de bruit. D'un autre côté, le silence est également réduit si tous les serveurs qui contiennent des documents pertinents sont sélectionnés.

TAB. 4.7 – *Comparaison des deux approches de sélection PasSel et Optimal1 combinées avec différentes approches de fusion.*

	Environnement			
	[(TREC8.1 , Okapi), ⋮ (TREC8.4 , Okapi)]		[(TREC9.1 , Okapi), ⋮ (TREC9.8 , Okapi)]	
Système	*PM*	diff.	*PM*	diff.
(*PasSel*, *RSM*)	0.2397		0.1832	
(*Optimal1*, *RSM*)	0.2543	+5.68%	0.2097	+13.52%
(*PasSel*, *CoriF*)	0.2416		0.1847	
(*Optimal1*, *CoriF*)	0.2533	+4.55%	0.2142	+15.05%
(*PasSel*, *LMS*)	0.2462		0.1932	
(*Optimal1*, *LMS*)	0.2480	+0.70%	0.2144	+10.81%

D'après les résultats présentées dans la table 4.7, nous constatons une augmentation de performance lorsque *Optimal1* est utilisée et ce quelque soit la méthode de fusion utilisée. Néanmoins cette augmentation n'est pas significative dans tous les cas. Nous remarquons, en effet, les deux phénomènes suivants :

- Lorsque l'on utilise une méthode de fusion peu efficace (*RSM*), *Optimal1* améliore significativement la performance comparée à *PasSel*, dans le cas de TREC9, par exemple, cette amélioration est même importante (+13.52%)). Cependant, en présence d'une méthode de fusion efficace (*LMS*), la différence entre les deux approches *PasSel* et *Optimal1* est moins importante. Ceci peut s'expliquer par le fait que (*PasSel*, *LMS*) soit plus performante que (*PasSel*, *RSM*). Nous pouvons donc dire que l'utilisation d'une méthode de fusion efficace apporte une partie de l'amélioration de la performance d'un SRID (cas des systèmes utilisant *LMS*), et effectuer une *bonne* sélection apporte plus de performance (cas des systèmes utilisant *Optimal1*) ;
- L'efficacité de *Optimal1* est plus visible sur TREC9 que sur TREC8. Nous pensons que ceci est dû au fait que le corpus TREC9 contient plus de collections (huit) que TREC8 qui ne contient que quatre collections. En effet, lorsque le nombre de collection diminue le risque d'erreur de l'approche *PasSel* diminue aussi.

4.4.5 Notre méthode a-t-elle un fondement dans le cas idéal?

4.4.5.1 But

Les approches de sélection de serveurs que nous exposons dans cette thèse sont basées sur l'hypothèse suivante : il ne suffit pas qu'un serveur contiennent des documents pertinents pour être utile, il faudra en plus qu'il soit capable de les

retourner. Nous, avons choisi de vérifier l'utilité d'un serveur en analysant les premiers documents qu'il retourne. Si un serveur présente des documents pertinents parmi les premiers documents qu'il retourne, cela signifie qu'il contient des documents pertinents et qu'il est capable de les retourner.
Notre but est de vérifier cette hypothèse sur nos corpus. En d'autres termes, nous voulons vérifier si, dans le cas où nous avons la connaissance exacte sur la pertinence des documents, notre approche est valide. A cet effet, nous décidons de définir une approche *Optimal2* pour tester cette hypothèse. *Optimal2* utilise les jugements de pertinence qui accompagnent les corpus afin de sélectionner les serveurs qui retournent au moins un document pertinent parmi les *nd* premiers retournés. Dans notre expérimentation nous faisons varier *nd* entre un et cinq.

4.4.5.2 Description

Cette expérimentation se déroule comme suit :

Pour chaque environnement
*[(*TREC8*.1 , Okapi),..,(*TREC8*.4 , Okapi)] et*
*[(*TREC9*.1 , Okapi),..,(*TREC9*.8 , Okapi)]* :
- *Pour nd dans {1, 2, 3, 4, 5}* :
  - *Pour chaque requête-test q* :
    - déterminer les serveurs qui retournent au moins un document pertinent parmi ses *nd* premiers résultats en se basant sur les jugements de pertinence accompagnant la collection-test ;
    - interroger à l'aide de Smart les serveurs sélectionnés ;
    - fusionner les résultats à l'aide de *RSM* ;
    - appliquer *trec_eval* afin de calculer PM_q la précision moyenne des résultats .
  - Calculer la moyenne PM des PM_q

4.4.5.3 Résultats

Les résultats de cette expérimentation sont présentés dans la table 4.8, nous y indiquons la moyenne des PM obtenues lorsque au moins un serveur est sélectionné pour une requête. Nous avons constaté que pour de nombreuses requêtes aucun serveur n'est sélectionné. Ainsi, nous montrons dans la deuxième colonne le nombre de requêtes prises en compte pour chaque ligne correspondante du tableau. En outre, nous mettons entre parenthèses le pourcentage de la différence de performance par rapport à l'approche centralisée. La dernière colonne représente la différence entre les apports de *Optimal1* et de *Optimal2* par rapport à l'approche centralisée.

TAB. 4.8 – *Comparaison des deux approches Optimal1 et Optimal2, toutes les deux combinées à RSM.*

environnement	*nd*	Nb. Req.	*Centr*	(*Optimal1*, *RSM*)	(*Optimal2*, *RSM*)	Diff.
[(TREC8.1 , Okapi),	1	37	0.3050	0.3164 (+3.7%)	0.3131 (+2.6%)	-1.28%
.	2	46	0.2655	0.2736 (+3.0%)	0.2646 (-0.2%)	-3.50%
.	3	46	0.2655	0.2736 (+3.0%)	0.2700 (+1.7%)	-1.40%
,(TREC8.4 , Okapi)]	4	48	0.2582	0.2660 (+3.0%)	0.2635 (+2.0%)	-0.97%
	5	49	0.2542	0.2616 (+2.9%)	0.2625 (+3.2%)	+0.35%
[(TREC9.1 , Okapi),	1	32	0.2864	0.3044 (+6.3%)	0.2929 (+2.3%)	-5.86%
.	2	32	0.2864	0.3044 (+6.3%)	0.3099 (+8.2%)	+2.8%
.	3	35	0.2501	0.2630 (+5.2%)	0.2826 (+13.0%)	+10%
,(TREC9.8 , Okapi)]	4	38	0.2462	0.2640 (+7.2%)	0.2727 (+10.8%)	+4.43%
	5	39	0.2402	0.2579 (+7.4%)	0.2717 (+13.1%)	+7.04%

4.4.5.4 Discussion

Les résultats que nous avons obtenus sur le corpus TREC9 confirment notre hypothèse. Effectivement, à part pour le cas $nd = 1$, tous les résultats de l'approche *Optimal2* sont significativemement supérieurs aux deux approches *Centr* et *Optimal1*. En d'autres termes, il est plus judicieux de sélectionner les serveurs sur la base des documents pertinents qu'ils **retournent**, plutôt que de sélectionner les serveurs sur la base de documents pertinents qu'ils **contiennent**. En revanche, sur le corpus TREC8, la différence de performance entre les deux approches *Optimal1* et *Optimal2* n'est pas significative. Nous pensons que la similitude des performances de ces deux approches, dans l'environnement TREC8, est due au nombre réduit de serveurs de ce dernier. En effet, ceci réduit le risque d'erreur qu'un serveur sélectionné par *Optimal1*, donc contenant au moins un document pertinent, ne soit sélectionné par *Optimal2*.

4.4.6 L'approche *CORI*

4.4.6.1 But

Dans cette expérimentation, nous désirons analyser la performance de l'approche *CORI* sur nos collections et la combiner avec différentes méthodes de fusion. Le but est de pouvoir comparer par la suite les résultats que nous obtenons en utilisant nos approches de sélection à la performance du système *CORI*.

4.4.6.2 Description

L'expérimentation est effectuée comme suit :

Pour chaque environnement
*[(*TREC8*.1 , Okapi),..,(*TREC8*.4 , Okapi)] et*
*[(*TREC9*.1 , Okapi),..,(*TREC9*.8 , Okapi)]* :

- *Pour chaque requête-test q* :
  - - Déterminer les serveurs à interroger à l'aide de *CORI* ;
    - interroger à l'aide de Smart les serveurs sélectionnés ;
    - *Pour tout algorithme* **Fus** *parmi les algorithmes de fusion RSM, LMS et CoriF* :
      - - fusionner les résultats par **Fus** ;
        - appliquer *trec_eval* afin de calculer PM_q la précision moyenne des résultats .
- Calculer la moyenne PM des PM_q

4.4.6.3 Résultats

Le tableau 4.9 indique les résultats obtenus par cette expérimentation. Les valeurs entre parenthèses déterminent la différence entre l'approche en question et l'approche centralisée.

TAB. 4.9 – *Approche CORI: PM des résultats retournés par les systèmes (CORI, RSM),(CORI, CoriF) et (CORI, LMS)*

	Environnement	
Système	[(TREC8.1 , Okapi), ⋮ (TREC8.4 , Okapi)]	[(TREC9.1 , Okapi), ⋮ (TREC9.8 , Okapi)]
(CORI, *RSM*)	0.20005 (-21.86%)	0.1862 (-5%)
(CORI, *CoriF*)	0.2033 (-20.77%)	0.1893 (-3.41%)
(CORI, *LMS*)	0.1997 (-22.17%)	0.1922 (-1.93%)

4.4.6.4 Discussion

Lorsque *CORI* est appliquée à TREC8, nous constatons clairement une performance moindre. En effet, quelque soit la méthode de fusion avec laquelle *CORI* est combinée, les résultats restent médiocres soit approximativement 20% de performance en moins par rapport à l'approche centralisée.
Les résultats dans TREC9 sont meilleurs. Dans cet environnement, le système (*CORI*, *RSM*) est le moins performant par rapport aux deux autres, à savoir (*CORI*, *CoriF*) et (*CORI*, *LMS*). Les résultats de ces deux derniers systèmes ne présentent pas une différence significative avec les résultats du système centralisé.

4.4.7 Nos approches de sélection de serveurs

4.4.7.1 But

Le but des différentes expérimentations qui vont suivre est d'analyser les performances obtenues en utilisant nos approches de sélection combinées aux approches de fusion *RSM* et *LMS*.

4.4.7.2 Description

Avant de pouvoir tester sur un corpus l'une de nos approches de sélection de serveurs, nous devons donner des valeurs aux paramètres utilisés. Pour cela, nous définissons, dans un premier temps, un espace de variation pour chaque paramètre, nous effectuons ensuite les tests pour chaque combinaison de valeurs de ces paramètres prises dans cet espace. Cette dernière opération nous permettra, d'une part, d'analyser la performance de nos approches, et d'autre part, de déterminer les valeurs optimales des paramètres permettant la meilleure performance.

Déterminer les espaces de variation des paramètres de nos approches

Nous manipulerons deux types de paramètres : les paramètres des approches (*param_appr*) et les paramètres des tests (*param_test*). Les *param_appr* sont les paramètres utilisés par nos approches de sélection comme, par exemple, le nombre de serveur à sélectionner ou la longueur de la liste de résultats à retourner à l'utilisateur. Les *param_test* sont ceux utilisés pendant un test comme, par exemple, la méthode de sélection ou la méthode de fusion à tester.

Dans ce qui suit, nous rappelons étape par étape le processus de RID effectué dans le cadre de nos approches, tout en extrayant les paramètres utilisés.

1. Le courtier commence par l'étape d'acquisition des représentants des serveurs. Dans cette étape, un score est calculé pour chacun des k premiers documents retournés par chacun des serveurs du système. Pour ce faire, nous avons besoin de définir les valeurs des paramètres suivants :

   (a) A : le paramètre qui détermine la distance maximale tolérée entre deux termes de la requête dans un document. Nous avons choisi quatre valeurs de test pour A :

   - 0 : une valeurs *restrictive*. Nous supposons que deux termes de la requête doivent apparaître côte à côte dans un document pour attribuer à ce dernier le score maximal (1) ;
   - 8 : la distance entre les deux mots limitant une phrase de longueur usuelle, i.e. une phrase de 10 mots (valeur correspondant à notre capacité moyenne de rétention immédiate des informations) ;
   - 16 : est la valeur utilisé dans [CCB95] qui décrit la méthode dont on s'est inspiré afin de prendre en compte la distance entre les termes dans le calcul du score d'un document ;
   - 100 : une valeur *tolérante* ;

   (b) c_1, c_2, c_3 sont des constantes qui déterminent le degré d'importance de chaque élément utilisé dans le calcul du score d'un document d_{ij}. Ces éléments sont, respectivement, nb_terme_{ij}, le nombre de termes de la requête apparaissant dans d_{ij}, nb_occ_{ij}, le nombre de fois où ces termes apparaissent dans d_{ij} et ind_dis_{ij} l'indicateur de la distance entre les deux premiers termes de la requête dans d_{ij}.

   Normalement, nous devrions chercher les valeurs relatives idéales de c_1, c_2 et c_3. Or, cette recherche s'avère, en pratique coûteuse en temps. Nous nous sommes alors limités à déterminer, par le biais de ces constantes, l'importance booléenne[5] de chaque élément suscité plutôt que le degré d'importance de ces éléments. Vu que les éléments nb_terme_{ij}, nb_occ_{ij} et ind_dis_{ij} appartiennent, respectivement, aux intervalles de valeurs suivants : $[0,3] \subset \mathbb{N}$, $[0,6]$[6]$\subset \mathbb{N}$

5. i.e. si oui ou non un élément est important

6. la valeur 6 étant la moyenne des occurrences d'un terme dans un document dans TREC9, cette valeur est de 21 dans TREC8

et $[0,1] \subset \mathbb{R}$, nous choisissons de tester les valeurs arbitraires de 1 et 1000 pour les paramètres c_1, c_2 et c_3. Ainsi, par exemple, pour la configuration $(1, 1000, 1000)$ de (c_1, c_2, c_3), nous considérons que l'élément nb_terme_{ij} doit être negligé dans le calcul du score d'un document d_{ij}.

(c) k : le nombre de documents à analyser parmi les documents les mieux classés retournés par chacun des n serveurs du système. Nos approches supposent le chargement de $n \times k$ documents par le courtier. Par souci d'économie nous pensons que k ne devrait pas être supérieur à 5 car le temps de transfert pourrait alors pénaliser le système. Nous avions donc l'intention de varier k entre 1 et 5. Cependant, nous n'avons effectué des tests que sur le cas $k = 3$. Nous expliquerons à la fin de cette section les raisons de cette limitation. Nous testerons les systèmes en variant le paramètre k une fois les valeurs optimales des autres paramètres fixés.

2. Le courtier calcule le score de chaque serveur S_i selon l'une des approches appelées $Score_1(S_i), \ldots, Score_5(S_i)$. Ces approches reposent, respectivement, sur les hypothèses H1, H2, H3, H4 et H5 décrites dans la section 3.4.3 du chapitre 3. Nous définissons donc un *param_test*, nommé H, dont la valeur correspond à 1, 2, 3, 4 ou 5 selon l'hypothèse utilisée.

3. Le courtier effectue la sélection en se basant sur les scores des serveurs calculés pendant la précédente étape. Nous avons proposé dans cette thèse trois types de sélection à savoir, *CS-SNF*, *CS-SS* et *SNB*. Nous définissons un *param_test* que nous nommons *TypeSel* et dont la valeur correspond au nom de la méthode de sélection utilisée.

   Ces trois types de sélection utilisent toutes un paramètre, *nd*, représentant le nombre de documents qui sont considérés pertinents par le courtier parmi les $n \times k$ documents chargés. Ces *nd* documents serviront à choisir les serveurs à sélectionner. Nous avons choisi de tester nos approches pour les valeurs de *nd* permettant de donner, dans le cas restrictif, une chance à 50% des serveurs d'être sélectionnés et, dans le cas tolérant, la possibilité d'éliminer au maximum 25% des serveurs. D'autres valeurs intermédiaires, arbitrairement choisies, entre les deux valeurs restrictive et tolérante de *nd* sont aussi testées pour le paramètre *nd*.

   Il reste à définir les valeurs des paramètres spécifiques à chaque type de sélection à savoir :

   - *TypeSel*= *CS-SNF* : dans ce cas le courtier sélectionne les *ns* serveurs les mieux classés. Nous varions *ns* de 1 à n.
   - *TypeSel*= *CS-SS* : le courtier sélectionne les serveurs dont le score dépasse un certain seuil *sl*. La valeur choisie pour *sl* dépend de l'approche utilisée pour calculer le score d'un serveur ($Score_1(S_i)$, ..., $Score_5(S_i)$).

     Pour chacune des approches $\{Score_1(S_i), \ldots, Score_5(S_i)\}$, on définit

une valeur *restrictive*, une valeur *tolérante* et trois valeurs intermédiaires pour le paramètre *sl*.
La valeur *tolérante* correspond au cas où le courtier sélectionne tout serveur retournant au moins un document pertinent. La valeur *restrictive* dépend de l'approche de calcul de score dont il est question. Nous définissons donc pour chacune des approches de calcul du score d'un serveur le quintuplet

$$(Restr, \frac{Rest + (3 \cdot Tol)}{4}, \frac{(2 \cdot Rest) + (2 \cdot Tol)}{4}, \frac{(3 \cdot Rest) + Tol}{4}, Tol)$$

tel que $Restr$ est la valeur restrictive de sl et Tol la valeur tolérante. En notant Sc_i le score d'un document d_i appartenant à $DocChar_{nd}$, nous avons calculé les valeurs *tolérante* et *restrictive* comme suit :

- L'approche $Score_1(S_i)$ considère le nombre de documents pertinents retournés par un serveur S_i. Nous choisissons donc $Restr = k$ et $Tol = 1$.
  Dans le cas *restrictif*, le serveur sélectionné correspond à celui dont les k documents retournés sont tous pertinents.
- L'approche $Score_2(S_i)$ prend en compte les scores des documents pertinents qu'un serveur S_i retourne. Nous choisissons :

  $$Restr = \sum_{i=1}^{i=k} Sc_i$$

  et

  $$Tol = \min_{1 \leq i \leq nd} Sc_i$$

  Dans le cas *restrictif*, un serveur est sélectionné, si les k documents qu'il retourne correspondent aux k meilleurs document de $DocChar_{nd}$.
- L'approche $Score_3(S_i)$ considère le score maximal des documents qu'un serveur retourne afin d'attribuer un score à ce serveur. Dans ce cas nous optons pour :

  $$Restr = \max_i Sc_i$$

  et

  $$Tol = \min_i Sc_i$$

  Le seul serveur sélectionné dans le cas *restrictif* est le serveur du document dont le score est le maximum parmi les documents $DocChar_{nd}$.
- L'approche $Score_4(S_i)$ considère le score moyen des documents pertinents d'un serveur. Nous proposons alors :

  $$Restr = \sum_{i=1}^{k} Sc_i/k$$

et

$$Tol = \min_i Sc_i$$

Dans le cas *restrictif*, un serveur est sélectionné, si les k documents qu'il retournent correspondent aux k meilleurs documents de $DocChar_{nd}$.

- L'approche $Score_5(S_i)$ donne de l'importance au nombre de documents pertinents retournés par un serveur et aussi au score le plus élevé de ces derniers. Nous adoptons alors :

$$Restr = k \cdot \max_i Sc_i$$

et

$$Tol = \min_i Sc_i$$

Dans ce cas, le seul serveur sélectionné est celui dont tous les documents retournés sont pertinents et l'un de ses documents a le score le plus élevé parmi les documents de $DocChar_{nd}$.

- $TypeSel$= SNB : le courtier détermine le nombre de documents, retournés par un serveur S_i, à inclure dans la liste finale de taille L. Nous avons choisi de tester l'approche SNB pour les valeurs 5, 10, 100 et 1000 du paramètre L. Nous avons choisi les valeurs 5 et 10 car nous pensons que l'utilisateur consulte en général les 5 à 10 premiers documents retournés. Les autres valeurs ont été arbitrairement choisies afin de comparer avec les cas des deux valeurs précédentes.

4. Le courtier effectue la fusion des résultats. Nous avons décidé de tester deux procédures de fusion à savoir RSM et LMS. Nous définissons donc un $param\_test$ que nous nommons Fus et qui va correspondre à la méthode de fusion utilisée.

La table 4.10 récapitule les espaces de variation que nous venons de définir.

TAB. 4.10 – *Les différents paramètres utilisés dans les expérimentations et leurs valeurs testées*

Paramètres des approches (*param_appr*)		
A	Paramètre de distance, intervenant dans le calcul du score d'un document au niveau du courtier.	0, 8, 16, 100
$c1$, $c2$, $c3$	Paramètres de pondération utilisés dans le calcul du score d'un document au niveau du courtier.	1, 1000
k	Nombre de documents que le courtier analyse afin de juger de la pertinence des serveurs.	1, 2, **3**, 4, 5
nd	Nombre de documents de $DocChar_{nd}$ jugés pertinents par le courtier.	Pour TREC9 → 4, 8, 10, 16, 18, 19, 20, 22
ns	Nombre de serveurs à sélectionner (cas de la sélection *CS-SNF*).	Pour TREC9 → 1, . . . ,8
sl	Score minimal des serveurs qui seront sélectionnés (cas de la sélection *CS-SS*).	$Restr$, $\frac{Rest+(3\cdot Tol)}{4}$, $\frac{(2\cdot Rest)+(2\cdot Tol)}{4}$, $\frac{(3\cdot Rest)+Tol}{4}$, Tol
L	Longueur de la liste finale des résultats à présenter à l'utilisateur (cas de la sélection *SNB*).	5, 10, 20, 100, 1000
Paramètres des tests (*param_test*)		
H	Numéro de l'hypothèse sur laquelle repose la formule utilisée dans le calcul du score d'un serveur.	1, 2, 3, 4, 5
$TypeSel$	Type de sélection de serveur utilisée	*CS-SNF*, *CS-SS*, *SNB*
Fus	Méthode de fusion utilisée	*RSM*, *LMS*
$Envir$	Environnement des tests.	[(TREC9.1, Okapi), ... ,(TREC9.8, Okapi)] ⋮ [(TREC8.1, Okapi), ... ,(TREC8.4, Okapi)]

Nous limitons nos évaluations à l'environnement [(TREC9.1,Okapi), · · · ,(TREC9.8,Okapi)]. Les évaluations dans TREC8 concerneront seulement les valeurs optimales des paramètres. De plus, nous avons limité le paramètre k à la valeur 3 à cause du manque de temps. En effet, il aurait fallu exécuter la procédure de sélection et de fusion 276 480 fois au lieu de 27 648 fois, afin d'analyser en détails les variations possibles des paramètres k et $Envir$. Sachant que la durée moyenne d'une itération (sélection et fusion) est de trois minutes, il aurait fallu 576 jours

de traitement sur une machine unique (Sparc ULTRA de 128 méga-octets de mémoire centrale et un processeur à 333 MHZ).

Les tests

Pour notre analyse, nous avons lancé la procédure de sélection suivie de la procédure de fusion pour chaque combinaison des valeurs des paramètres spécifiés ci-dessus. Le processus se décrit par l'algorithme suivant :

*Pour l'environnement ([(*TREC9*.1 , Okapi),..,(*TREC9*.8 , Okapi)]) et pour k=3*
- *Pour (nd = 4, 8, 10, 16, 18, 19, 20, 22)*
  - *Pour (A = 0, 8, 16, 100)*
    - *Pour (c1 = 1, 1000)*
      - *Pour (c2 = 1, 1000)*
        - *Pour (c3 = 1, 1000)*
          - *Pour (H = 1, 2, 3, 4, 5)*
            - *Pour (TypeSel = SNB, CS_SNF, CS_SS)*

              Si (TypeSel = SNB)
              - *Pour (L = 5, 10, 100, 1000)*
                - *Pour chaque requête-test q*
                  - Effectuer la sélection SNB.
                  - *Pour (Fus = RSM, LMS)*
                    - –Fusionner les résultats de la sélection avec Fus.
                    - –Appliquer trec_eval afin de calculer PM_q la précision moyenne moyenne des résultats de la fusion.
                - Calculer la moyenne PM_{Fus} des PM_q.

              Si (TypeSel = CS-SNF)
              - *Pour (ns = 1, 2, 3, 4, 5)*
                - *Pour chaque requête-test q*
                  - Effectuer la sélection *CS-SNF*.
                  - *Pour (Fus = RSM, LMS)*
                    - –Fusionner les résultats de la sélection avec Fus.
                    - –Appliquer trec_eval afin de calculer PM_q la précision moyenne moyenne des résultats de la fusion.
                - Calculer la moyenne PM_{Fus} des PM_q.

              Si (TypeSel = CS-SS)
              - *Pour (seuil = Restr,* $\frac{Rest+Tol}{4}$, $\frac{Rest+Tol}{2}$, $\frac{3*(Rest+Tol)}{4}$, *Tol)*
                - *Pour chaque requête-test q*
                  - Effectuer la sélection *CS-SS*.
                  - *Pour (Fus = RSM, LMS)*
                    - –Fusionner les résultats de la sélection avec Fus.
                    - –Appliquer trec_eval afin de calculer PM_q la précision moyenne moyenne des résultats de la fusion.
                - Calculer la moyenne PM_{Fus} des PM_q.

4.4.7.3 Résultats

Les résultats de cette expérimentation sont présentés dans la table 4.11. Les valeurs entre parenthèses déterminent le pourcentage de la différence entre la performance de nos approches

et celle de l'approche centralisée.
En étudiant l'intégralité des résultats obtenus, nous avons constaté que dans 9205 cas (soit 33% des cas testés) nos approches ne retournent pas systématiquement une réponse à chaque requête-test. La cause est certainement due au cas des valeurs restrictives des paramètres. Cette absence de réponse rend la comparaison de nos approches avec d'autres impossible.
Les résultats présentés dans le tableau 4.11 regroupent les meilleurs résultats atteints par nos approches. c'est-à-dire, ayant une bonne précision moyenne, tout en sélectionnant un nombre raisonnable de serveurs (dans nos cas 6). Un nombre raisonnable de serveurs sélectionnés est un nombre pas trop grand afin de réduire les coûts d'interrogation, et pas trop petit afin d'assurer une bonne performance du système.

TAB. 4.11 – *Les meilleurs résultats obtenus lors de la variation des paramètres de nos approches, dans l'environnement [(*TREC9*.1, Okapi), ... ,(*TREC9*.8, Okapi)].*

Système	Valeurs des paramètres							PM
	nd	*A*	c_1	c_2	c_3	*H*	-	
(*CS-SNF*, *RSM*)	16	100	1000	1	1000	2,5	*ns* =6	0.1825 (-6.8%)
(*CS-SNF*, *LMS*)	16	100	1000	1	1000	2,5	*ns* =6	0.1940 (-1.02%)
(*CS-SS*, *RSM*)	19	16	1	1	1	1-5	*sl* =1	0.1822 (-7%)
(*CS-SS*, *LMS*)	19	16	1	1	1	1-5	*sl* =1	0.1938 (-1.12%)
(*SNB*, *RSM*)	19	16	1	1	1	1,2,4	*L* =1000	0.1823 (-6.9%)
(*SNB*, *LMS*)	19	16	1	1	1	1,2,4	*L* =1000	0.1234 (-37%)

4.4.7.4 Discussion

Discussion sur les valeurs optimales des paramètres

Pour nos trois approches de sélection, les deux valeurs optimales de *nd* sont 16 et 19. Ces deux valeurs sont celles qui doublent la chance qu'un serveur soit sélectionné. En effet, un serveur répond par trois (dans notre cas k=3) documents, il possède donc trois chances d'être sélectionné (car sa sélection dépend de ces documents).
Concernant le paramètre *A*, indiquant la distance maximale tolérée entre les occurrences des deux premiers termes de la requête dans un document, les valeurs optimales sont 16 et 100. Ceci était prévisible car les deux autres valeurs (0 et 8) de *A* sont trop restrictives. Néanmoins, 100 étant une valeur tolérante de *A*, il nous semble donc que pour l'approche *CS-SNF*, l'indicateur de distance entre les occurences des deux premiers termes de la requête ne joue qu'un rôle secondaire, bien que la valeur optimale de $c3$ porte à croire que cet indicateur est important dans le calcul du score d'un document.

En examinant les valeurs optimales des constantes $c1$, $c2$ et $c3$, qui indiquent l'importance des éléments *nb_terme*, *nb_occ* et *ind_dis* (voir 3.4.2.3 du cha-

pitre 3), nous déduisons que ces derniers sont tous relativement importants selon la méthode de sélection utilisée. En effet :

- pour l'approche *CS-SNF*, les valeurs de ces constantes semblent indiquer que l'élément *nb_occ* n'est pas important dans le calcul du degré de pertinence d'un document, alors que les deux éléments *nb_terme* et *ind_dis* sont importants dans ce calcul. Néanmoins, et comme nous l'avons mentionné ci-dessus, l'importance de *ind_dis* est négligeable car la distance tolérée entre les termes est large (A =100) ;

- pour les deux approches *CS-SS* et *SNB*, les trois éléments sont importants car la configuration optimale de ($c1$,$c2$,$c3$) est de (1,1,1).

Pour ce qui est du paramètre H, l'approche *CS-SNF* fonctionne mieux avec les deux hypothèses H2 et H5 et l'approche *SNB* avec les hypothèses H1, H2 et H4 (synthèse de H1 et H2). En revanche, *CS-SS* est insensible aux différentes hypothèses considérées. Une des explications que nous pouvons donner est que *CS-SS* ne possède pas de paramètre spécifique fixe comme c'est le cas de *CS-SNF* et *SNB* qui fixent respectivement, les paramètres *ns* et L quelque soit H. En effet, *CS-SS* utilise le paramètre *sl* qui dépend de la formule de calcul du score d'un serveur et donc de l'hypothèse considérée, en d'autres termes, *sl* est calculé en fonction de H.

Discussion sur la performance de nos approches

Les résultats de la table 4.11 indiquent que nos trois approches, lorsqu'elles sont combinées à la méthode de fusion *RSM*, atteignent approximativement la même performance. Cette performance n'est pas satisfaisante comparée à l'approche centralisée. En effet, nos approches présentent une dégradation de performance de l'ordre de 7% par rapport à la performance de l'approche centralisée.
En revanche, nos deux approches *CS-SNF* et *CS-SS*, combinées avec la méthode de fusion *LMS*, donnent de bons résultats avec des précisions moyennes égales à la précision moyenne de l'approche centralisée. En effet, nos deux systèmes (*CS-SNF*, *LMS*) et (*CS-SS*, *LMS*) présentent, respectivement, (-1.02%) et (-1.12%) de différence par rapport au système *Centr*. Or, ces différences étant inférieures à 5%, elles ne peuvent être considérées comme significatives.
L'approche *SNB*, combinée à l'approche *LMS*, présente des résultats décevants avec une importante dégradation de performance (-37%) par rapport à l'approche centralisée. Cette approche s'avérant peu intéressante quelque soit la méthode de fusion utilisée, nous avons donc décidé de l'écarter de nos prochaines expérimentations.

4.4.8 Nos approches : Tests supplémetaires

4.4.8.1 Déterminer la valeur optimale du paramètre k

4.4.8.1.1 But Le but de l'expérimentation suivante est de déterminer la valeur optimale de k, autrement dit, le nombre optimal des documents qu'un serveur doit retourner, pour une requête donnée. Tels documents vont servir à juger de la pertinence des serveurs.

4.4.8.1.2 Description Dans la section précédente (4.4.7) nous avons varié plusieurs paramètres et nous avons appliqué chaque combinaison de valeurs de ces paramètres à l'environnement [(TREC9.1, Okapi), ... ,(TREC9.8, Okapi)]

Pour chaque combinaison de valeurs de ces paramètres, nous avons appliqué à l'environnement [(TREC9.1 , Okapi), ... ,(TREC9.8 , Okapi)], nos trois approches de sélection de serveurs combinées avec deux approches de fusion. Nous en avons alors déduit les valeurs optimales de ces paramètres pour nos deux approches *CS-SNF* et *CS-SS*. En outre, nous avons écarté l'approche *SNB* qui n'apporte pas une performance intéressante. Dans les évaluations décrites dans la table 4.12, nous fixons les valeurs des paramètres aux valeurs optimales et nous varions les valeurs du paramètre k.

4.4.8.1.3 Résultats La table 4.12 montre les résultats que l'on obtient en variant k. Les valeurs entre parenthèses correpondent aux pourcentages de la différence de performance entre nos approches et l'approche centralisée.

TAB. 4.12 – *Les résultats obtenus par la variation du paramètre k dans l'environnement [(*TREC9*.1, Okapi), ... ,(*TREC9*.8, Okapi)].*

Système	k	PM	Système	k	PM
(*CS-SNF*, *RSM*)	1	0.1685 (-14,03%)	(*CS-SNF*, *LMS*)	1	0.1788 (-8,78%)
(*CS-SNF*, *RSM*)	2	0.1768 (-9,80%)	(*CS-SNF*, *LMS*)	2	0.1879 (-4,13%)
(*CS-SNF*, *RSM*)	3	0.1825 (-6,89%)	(*CS-SNF*, *LMS*)	3	0.1940 (-1,02%)
(*CS-SNF*, *RSM*)	4	0.1788 (-8,78%)	(*CS-SNF*, *LMS*)	4	0.1898 (-3,16%)
(*CS-SNF*, *RSM*)	5	0.1757 (-10,36%)	(*CS-SNF*, *LMS*)	5	0.1870 (-4,59%)
(*CS-SS*, *RSM*)	1	0.1810 (-7,65 %)	(*CS-SS*, *LMS*)	1	0.1932 (-1,43%)
(*CS-SS*, *RSM*)	2	0.1810 (-7,65 %)	(*CS-SS*, *LMS*)	2	0.1932 (-1,43%)
(*CS-SS*, *RSM*)	3	0.1822 (-7,04%)	(*CS-SS*, *LMS*)	3	0.1938 (-1,12%)
(*CS-SS*, *RSM*)	4	0.1818 (-7,24%)	(*CS-SS*, *LMS*)	4	0.1931 (-1,48%)
(*CS-SS*, *RSM*)	5	0.1823 (-6,99%)	(*CS-SS*, *LMS*)	5	0.1936 (-1,22%)

4.4.8.1.4 Discussion L'étude des résultats obtenus lors de cette expérimentation nous conduit à deux constatations. Premièrement, ces résultats s'accordent avec les résultats de la section 4.4.7 démontrant que nos approches *CS-SNF* et *CS-SS* retournent de meilleurs résultats lorsqu'elles sont combinées avec la méthode de fusion *LMS* plutôt qu'avec la méthode *RSM*. Deuxièmement, nous constatons que *trois* semble être la valeur optimale du paramètre k. En effet, la précision moyenne obtenue lorsque k prend la valeur trois est supérieure (ou égale dans certains cas) à la précision moyenne des résultats obtenues pour les autres valeurs de k et ceci tant avec l'approche de sélection *CS-SNF* que *CS-SS* et pour n'importe quelle méthode de fusion.

4.4.8.2 Application des valeurs des paramètres optimaux à TREC8

4.4.8.2.1 But Le but de cette expérimentation est de vérifier la performance de nos approches dans une configuration différente de la précédente à savoir l'environnement [(TREC8.1 , Okapi), ... ,(TREC8.4 , Okapi)]. Cet environnement est différent par le nombre de serveurs sous-jacents d'une part, et d'autre part, par le fait que les documents contenus dans une même collection du présent environnement ont des chances d'être de contenus proches, puisque provenant d'une même source. Ce n'est pas le cas des documents d'une même collection de l'en-

vironnement TREC9, où, rappelons-le, les collections sont construites sur un critère différent. Ces collections sont construites de façon à ce qu'elles contiennent toutes, approximativement, le même nombre de documents.

4.4.8.2.2 Description Nous appliquons nos deux approches de sélection de serveurs (*CS-SNF* et *CS-SS*) combinée chacune avec les méthodes de fusion *RSM* et *LMS*. Concernant les valeurs des paramètres, nous avons adapté au nouvel environnement les valeurs optimales des paramètres déterminées précédemment. Cependant, nous avons gardé les même valeurs des paramètres que l'on considère indépendants de la taille des collections, à savoir, k, A, $c1, c2, c3$, H, sl. Nous avons changé les valeurs des deux paramètres nd et ns. Selon les approches *CS-SNF* ou *CS-SS*, la valeur optimale de nd sur le corpus TREC9 était de 16 ou 19 correspondant respectivement à approximativement 70% et 80% des documents chargés et classés par le courtier pour une requête donnée. Lorsque l'on est dans un environnement à quatre serveurs au lieu de huit, nous obtenons approximativement les valeurs 8 et 9 pour nd.

Dans le corpus TREC9, le paramètre ns prend la valeur 6, correspondant à 75% des serveurs, dans un environnement à quatre serveurs la valeur correspondante est 3.
L'expérimentation se déroule donc selon l'algorithme suivant :

*Dans l'environnement [(*TREC8*.1 , Okapi), ... ,(*TREC8*.4 , Okapi)]*
- *Pour TypeSel = (CS-SNF, CS-SS) et les valeurs suivantes des paramètres:*
  (k=3; nd =8; A = 100; (c1, c2, c3)=(1000, 1, 1000); H =2; ns =3),
  (k=3; nd =9; A = 16; (c1, c2, c3)=(1, 1, 1); H =1; sl =1),
  correspondant respectivement à CS-SNF et CS-SS:
  - *Pour chaque requête-test q :*
    - Effectuer la sélection avec $TypeSel$.
    - *Pour (Fus = RSM, LMS)*
      - −Fusionner les résultats de la sélection avec Fus.
      - −Appliquer trec_eval afin de calculer PM_q la précision moyenne des résultats de la fusion.
  - Calculer la moyenne PM des PM_q

4.4.8.2.3 Résultats La table 4.13 montre les résultats que nous obtenons en appliquant nos approches de sélection *CS-SNF* et *CS-SS* sur les documents de TREC8. Les valeurs entre parenthèses correpondent aux pourcentages de la différence de performance entre nos approches et l'approche centralisée.

TAB. 4.13 – *Les résultats de nos approches obtenus lors de l'application des paramètres optimaux dans l'environnement [(*TREC8*.1 , Okapi), ... ,(*TREC8*.4 , Okapi)].*

Système	Valeurs des paramètres							PM
	nd	*A*	c_1	c_2	c_3	*H*	-	
(*CS-SNF*, *RSM*)	8	100	1000	1	1000	2	*ns* =3	0.2330 (-9.1%)
(*CS-SNF*, *LMS*)	8	100	1000	1	1000	2	*ns* =3	0.2409 (-6.1%)
(*CS-SS*, *RSM*)	9	16	1	1	1	1	*sl* =1	0.2421 (-5.6%)
(*CS-SS*, *LMS*)	9	16	1	1	1	1	*sl* =1	0.2459 (-4.1%)

4.4.8.2.4 Discussion Avec la collection-test TREC9, nous avons constaté que nos approches donnent de meilleurs résultats lorsqu'elles sont combinées avec la méthode de fusion *LMS*. Ce résultat est confirmé par nos évaluations faites sur la collection-test TREC8. Néanmoins, nous constatons que lorsque nos les deux systèmes (*CS-SNF*, *LMS*) et (*CS-SS*, *LMS*) sont comparées au système centralisé, (*CS-SS*, *LMS*) ne présente pas une différence significative (-4.1%) contrairement à (*CS-SNF*, *LMS*). L'explication pourrait être que l'approche *CS-SS* est plus souple en sélection (sélection des serveurs ayant retourné au moins un document pertinent) alors que *CS-SNF* sélectionne un nombre fixe de serveurs. En effet, *CS-SS* sélectionne en moyenne par requête plus de serveurs (3.54) que *CS-SNF* qui ne sélectionne que trois serveurs par requête. La moyenne des serveurs qui doivent être sélectionnés[7] par requête sur l'environnement actuel étant de 3.1, nous pensons que le fait de ne pas sélectionner un serveur pertinent s'avère plus néfaste (c'est le cas pour *CS-SNF* où la moyenne des serveurs sélectionnés est inférieur à 3.1) que de sélectionner des serveurs qui ne sont pas pertinents (avec l'approche *CS-SS* où la moyenne des serveurs sélectionnés est supérieure à 3.1). Ceci s'explique par le fait que les collections de TREC8 sont plus homogènes, ainsi lorsque nous écartons une collection pertinente, il est probable que nous écartions par la même occasion un nombre important de documents pertinents.
Mais d'une manière globale, on peut dire que *CS-SS* est adaptée à un environnement où les collections sont construites de documents homogènes.

4.4.8.3 Application des scores locaux vs. recalculer les scores

4.4.8.3.1 But Le but de cette expérimentation est de vérifier s'il est utile de recalculer, au niveau du courtier, les scores des documents de *DocChar*, ou d'utiliser les scores locaux fournis par les serveurs. Nous supposons bien entendu

7. Cette moyenne est obtenue à l'aide des jugements de pertinence fournis par le corpus TREC8

que ces scores sont relativement comparables parce-que les serveurs utilisent tous le même MRI, ce qui est le cas ici avec le modèle Okapi.

4.4.8.3.2 Description Nous gardons dans cette expérimentation l'environnement [(TREC9.1, Okapi), ... ,(TREC9.8, Okapi)] et nous combinons chacune de nos approches de sélection *CS-SNF* et *CS-SS* avec la méthode de fusion *RSM*. Le but étant de comparer la performance d'une approche utilisant les scores locaux avec celle d'une stratégie recalculant les scores. L'expérimentation est décrite en détail par l'algorithme suivant :

*Dans l'environnement [(*TREC9*.1 , Okapi),..,(*TREC9*.8 , Okapi)]*
- *Pour TypeSel = (CS-SNF, CS-SS) utilisant les scores locaux (scores Okapi)*
  - *Pour chaque requête-test q :*
    - Effectuer la sélection avec $TypeSel$.
    - Fusionner les résultats de la sélection avec RSM.
    - Appliquer trec_eval afin de calculer PM_q la précision moyenne des résultats de la fusion.
  - Calculer la moyenne PM des PM_q

4.4.8.3.3 Résultats Les résultats de cette expérimentation sont présentés dans la table 4.14, La dernière colonne de cette table représente la différence entre les apports, par rapport à l'approche centralisée, des approches utilisant les scores locaux et ceux des approches utilisant les scores recalculés. Les colonnes #serv. contiennent le nombre moyen de serveurs sélectionnés.

TAB. 4.14 – *La différence en PM entre nos approches de sélection CS-SNF et CS-SS utilisant les scores locaux et ces mêmes approches utilisant les scores re-calculés.*

Système	utilisation des scores recalculés		utilisation des scores locaux		diff.%
	PM	#serv.	*PM*	#serv.	
(*CS-SNF*, *RSM*)	0.1825	6	0.1720	6	-5.3%
(*CS-SS*, *RSM*)	0.1822	6	0.1812	7.3	-0.5%
(*CS-SS*, *RSM*)	0.1822	6	0.1695	6	-6.4%

4.4.8.3.4 Discussion Sur la base des performances décrites dans le tableau 4.14, nous pouvons faire les deux constatations suivantes :

- l'approche *CS-SS* atteint la même performance dans les deux cas d'utilisation avec les scores locaux ou avec les scores recalculés (deuxième ligne du tableau). Or, lorsque l'on examine le nombre de serveurs sélectionnés dans chaque cas, nous constatons que la comparaison est biaisée. En effet,

l'approche *CS-SS*, utilisant les scores locaux, sélectionne 7.3 serveurs en moyenne alors que *CS-SS* utilisée avec les scores recalculés n'en sélectionne que 6. Ainsi, nous avons présenté, dans la dernière ligne du tableau 4.14, les résultats obtenus lorsque *CS-SS*, utilisée avec les scores locaux, ne sélectionne que 6 serveurs au lieu de 7.3. Ce cas correspond à l'application de *CS-SS* avec le paramètre *sl* positionné à 2 et le reste des paramètres positionnés aux valeurs optimales ;

- pour le même nombre, en moyenne, de serveurs sélectionnés (6 serveurs), nos deux approches *CS-SNF* et *CS-SS* lorsqu'elles sont utilisées avec les scores locaux présentent des dégradations de performance significatives comparées aux résultats obtenus lorsque les scores sont recalculés. Le recalcul du score ne semble donc pas être une opération superflue.

Nous pensons, mis à part les résultats obtenus par cette expérimentation, qu'il est important de recalculer les scores des documents de *DocChar*, car les scores calculés par les serveurs ne sont pas comparables. En effet, ces scores dépendent de statistiques qui sont différentes d'une collection à une autre. De plus, les scores locaux ne sont pas toujours disponibles dans les réponses des serveurs.

4.4.8.4 Cas d'un SRID où cohabitent différents modèles de RI

4.4.8.4.1 But Le but de cette expérimentation est de déterminer si nos approches de sélection de serveurs sont performantes dans un environnement proche de la réalité. En d'autres termes, un environnement constitué de serveurs utilisant des SRI qualitativement différents. Nous cherchons donc à savoir si nos approches, comme la théorie le suppose, détectent réellement les serveurs ayant une performance moindre.

4.4.8.4.2 Description Nous ne saurons comparer les résultats de nos approches à ceux d'un système centralisé car ce dernier utilise un seul modèle RI alors que nous testons nos approches dans un environnement œuvrant avec différents MRI. Nous choisissons donc de comparer nos approches à l'approche *PasSel*. Ainsi, nous pouvons savoir si une sélection apporte une augmentation dans la performance du système. Nous analyserons aussi dans ce nouvel environnement l'approche du système *CORI*.
Pour cette expérience, nous avons choisi de tester nos approches dans un environnement constitué, d'une part, de très bons MRI (Okapi) et, d'autre part, des MRI ayant une performance médiocre soit nnn. Nous choisissons un nombre égal (quatre) de bons et de moins bons MRI.
Nous avons, également, choisi de tester nos approches pour des valeurs de paramètres qui réduisent le nombre de serveurs sélectionnés, afin de voir si les serveurs utilisant les bons MRI sont les mieux classés. Cette expérience se base sur *CS-SNF* avec $ns = 4$ et *CS-SS* avec $nd = 4$. A l'aide de ces valeurs, nous

donnons une chance pour chaque bon serveur d'être sélectionné, sachant que le système contient quatre bons serveurs.
Notre expérimentation est décrite par l'algorithme suivant :

*Dans l'environnement [(*TREC9*.1, nnn), (*TREC9*.2, Okapi), (*TREC9*.3, Okapi),(*TREC9*.4, nnn), (*TREC9*.5, nnn), (*TREC9*.6, nnn), (*TREC9*.7, Okapi),(*TREC9*.8 , Okapi)]*
- *Pour TypeSel = (CS-SNF, CS-SS, CORI, PasSel)*
  - *Pour chaque requête-test q :*
    - Effectuer la sélection avec $TypeSel$ avec TypeSel.
    - Fusionner les résultats de la sélection avec RSM.
    - Appliquer trec_eval afin de calculer PM_q la précision moyenne des résultats.
  - Calculer la moyenne PM des PM_q

4.4.8.4.3 Résultats Nous présentons dans la table 4.15 les PM obtenues par application des différentes approches *CS-SNF*, *CS-SS*, *PasSel*, *CORI*. Les valeurs entre parenthèses représentent la différence de PM de l'approche concernée avec l'approche *PasSel*, et la dernière colonne représente la moyenne du nombre de serveurs sélectionnés.

TAB. 4.15 – *Nos approches dans un environnement avec différents MRI: [(*TREC9*.1, nnn), (*TREC9*.2, Okapi), (*TREC9*.3, Okapi),(*TREC9*.4, nnn), (*TREC9*.5, nnn),(*TREC9*.6, nnn), (*TREC9*.7, Okapi),(*TREC9*.8 , Okapi)]*

Système	Valeurs des paramètres							PM	#serv.
	nd	A	c_1	c_2	c_3	H	-		
(*CS-SNF*, *RSM*)$_1$	16	100	1000	1	1000	2	ns =6	0.061 (+2.5%)	6
(*CS-SNF*, *LMS*)$_2$	16	100	1000	1	1000	2	ns =4	0.1096 (+27.2%)	4
(*CS-SS*, *RSM*)$_1$	19	16	1	1	1	1	sl =1	0.0560 (0%)	6
(*CS-SS*, *LMS*)$_2$	4	16	1	1	1	1	sl =1	0.1111 (+28.06%)	3
(*CORI*,*RSM*)								0.0562 (0%)	6.88
(*PasSel*, *RSM*)								0.0561	8

4.4.8.4.4 Discussion Considérons la performance des systèmes (*CS-SNF*, *RSM*)$_1$, (*CS-SS*, *RSM*)$_1$, (*CORI*,*RSM*) et (*PasSel*, *RSM*). Ces systèmes sélectionnent, en moyenne, respectivement 6, 6, 6.88, 8 serveurs pour chaque requête, sur un total de huit. Sachant qu'il existe dans notre environnement quatre serveurs utilisant le modèle nnn, il existe donc une chance réelle à ces derniers d'être sélectionnés. Les résultats dans ce cas sont médiocres (approximativement 5 de PM) ce qui était prévisible. En effet, sélectionner les serveurs dont le

MRI présente une performance médiocre engendre une augmentation du bruit dans la réponse finale.
Lorsque l'on oblige le courtier à sélectionner un nombre réduit de serveurs (ce nombre est fixé, dans notre expérimentation, au nombre de "bons" serveurs), on distingue une nette amélioration de performance par rapport à *PasSel*, de l'ordre de 27% et 28%, respectivement pour les approches *CS-SNF* et *CS-SS*. Ainsi, en diminuant le nombre de serveurs sélectionnés, la performance du système augmente. Ceci nous indique que les serveurs non sélectionnés étaient des serveurs non pertinents et donc porteurs de bruit. En effet, 70% des serveurs sélectionnés par nos deux approches *CS-SNF* et *CS-SS* correspondent à des serveurs utilisant le MRI Okapi. Nos approches arrivent donc bien à distinguer la performance relative des serveurs.

4.5 Comparaisons de différentes approches

4.5.1 Évaluation *EvalPerf*

Le tableau 4.16 (page 105) récapitule les résultats obtenus des évaluations des différentes approches présentées dans ce chapitre. Le but est de comparer nos deux approches *CS-SNF* et *CS-SS* avec les autres approches *Centr*, *PasSel* et *CORI*. Pour plus de clarté, nous adoptons pour l'interprétation des résultats de cette table les conventions suivantes :

- les valeurs en gras correspondent aux cas où la performance est significativement supérieure à la performance de l'approche centralisée ;
- les valeurs soulignées correspondent aux cas où la performance est significativement inférieure à la performance de l'approche centralisée ;
- enfin, les valeurs en style normal correspondent aux cas où la performance est égale à celle de l'approche centralisée.

Nous comparerons dans un premier temps les PM obtenus par ces approches. Dans un second temps, nous comparerons la $PM@x$ représentant la précision moyenne obtenue après x documents retrouvés. Cette mesure est utile car l'utilisateur typique s'intéresse et consulte souvent uniquement les premiers résultats [WSJS01].
Dans cette évaluation, nous ne considérons que les systèmes où la méthode de sélection est combinée à la méthode de fusion qui optimise le résultat final du système. Ainsi, *CORI* est combinée à *CoriF* dans TREC8 alors qu'elle est combinée à *LMS* dans TREC9. Les approches *PasSel*, *CS-SNF* et *CS-SS* sont elles, combinées à *LMS*.

TAB. 4.16 – *Récapitulatif des résultats obtenus par les différents systèmes testés.*

TREC8

Système	PM	PM@5	PM@10	PM@15	PM@20	PM@30	PM@100	Moyenne des serveurs sélectionnés
Centr	0.2566	0.4920	0.4560	0.4400	0.3970	0.3553	0.2280	
(*PasSel*, *LMS*)	0.2462	0.50	0.4380	0.4053	0.3820	0.3353	0.2146	4
(*CS-SNF*, *LMS*)	0.2409	0.5080	0.4520	0.4147	0.396	0.346	0.2146	3
(*CS-SS*, *LMS*)	0.2459	0.5080	0.4520	0.4133	0.39	0.34	0.2148	3.54
(*CORI*,*CoriF*)	0.2033	0.4880	0.4180	0.3787	0.3550	0.3093	0.1830	2.28

TREC9

Système	PM	PM@5	PM@10	PM@15	PM@20	PM@30	PM@100	Moyenne des serveurs sélectionnés
Centr	0.196	0.2667	0.2229	0.1917	0.1740	0.1597	0.1098	
(*PasSel*, *LMS*)	0.1932	0.2792	0.2208	0.1944	0.1760	0.1576	0.1075	8
(*CS-SNF*, *LMS*)	0.1940	**0.2917**	0.2333	**0.2056**	**0.1885**	**0.1694**	0.1042	6
(*CS-SS*, *LMS*)	0.1938	0.2792	0.2208	0.1944	0.1771	0.1604	0.1069	6
(*CORI*,*LMS*)	0.1922	0.2792	0.2250	0.1958	0.1771	0.1590	0.1065	7.52

La table 4.16 nous permet de faire plusieurs remarques :

1. Concernant le corpus TREC8 :
   - nos deux approches *CS-SNF* et *CS-SS* atteignent globalement des résultats satisfaisants. En effet, bien que *PM* pour l'approche *CS-SNF* soit légèrement inférieure à celle de *Centr*, lorsque nous considérons la précision des premiers résultats retournés à l'utilisateur (jusqu'à 30 documents), ces résultats sont égaux (la différence n'est pas significative) à ceux de l'approche *Centr* ;
   - lorsque nous comparons les résultats de *CS-SNF* et *CS-SS* aux résultats de *PasSel*, nous ne constatons pas de différence significative. Néanmoins, ces deux approches, à 30 documents, retournent des résultats proches de ceux de l'approche centralisée. En effet, à 30 documents, *CS-SNF* et *CS-SS* ne présentent pas de différence significative avec *Centr*, alors que *PasSel* présente une *PM*@30 inférieure à celle de *Centr* ;
   - l'approche *CORI* retourne des résultats insatisfaisants sauf dans le cas où nous considérons seulement les 5 premiers documents. Elle est moins performante que nos approches et s'avère moins performante que l'approche centralisée.
2. Concernant le corpus TREC9 :
   - nos deux approches *CS-SNF* et *CS-SS* retournent des résultats égaux à ceux obtenus par l'approche centralisée. Dans le cas de *CS-SNF*, les résultats sont supérieurs à ceux de *Centr* lorsque nous évaluons la performance sur la base de la précision obtenue après les premiers documents retournés (5, 15, 20 et 30 documents) ;
   - les résultats de nos approches comparés à ceux de l'approche *PasSel*, sont égaux (l'amélioration apportée par nos approches n'est pas significative). Cependant, lorsque l'évaluation de performance porte sur les premiers documents retournés (de 10 à 30 documents), *CS-SNF* est significativement supérieure à PasSel.
   - l'approche *CORI* fonctionne mieux dans l'environnement du corpus TREC9 que dans celui de TREC8. Les résultats retournés pas *CORI* dans TREC9 ne présentent pas de différence significative avec ceux de l'approche centralisée et ce quelque soit le nombre de documents pertinents retournés.

Dans l'ensemble, nous constatons que la performance de nos approches *CS-SNF* et *CS-SS* est satisfaisante comparée à celle de l'approche centralisée. La différence de précision est plus nette lorsque nous évaluons la pertinence des premiers documents retournés. Ces résultats sont aussi intéressants que ceux obtenus par l'approche *PasSel*. Nous avons donc réussi à obtenir, en ne sélectionnant qu'un certain nombre de serveurs, des résultats équivalents à ceux obtenus lorsque tous

les serveurs sont sélectionnés. Nous pouvons dire donc que nos approches sont performantes. Enfin, nous pouvons constater que nos approches fonctionnent aussi bien avec un nombre réduit de serveurs, comme c'est le cas dans l'environnement TREC8, qu'avec un nombre plus élevé de serveurs (TREC9).

4.5.2 Évaluation *EvalSel*

Dans la section précédente nous avons comparé les performances des différentes approches *Centr*, *PasSel*, *CORI*, *CS-SNF* et *CS-SS*. Cette comparaison se basait sur la précision moyenne qu'un système (méthode de sélection & méthode de fusion) pouvait obtenir. Bien que la satisfaction de l'utilisateur constitue notre premier objectif, nous avons tenu à évaluer nos approches en recourant aux méthodes d'évaluation du classement des serveurs. Le but de cette section est donc d'évaluer le classement, proprement dit, obtenu par nos deux approches à celui obtenu par *CORI*, *Optimal1* ou *CST*.
La méthode *CST* propose un classement constant des serveurs, en d'autres termes, un classement indépendant de la requête. Vu le caractère un peu arbitraire de cette approche, nous l'avons inclus dans nos évaluations afin de l'utiliser comme base de référence.
L'approche *Optimal1* représente l'approche réalisant le meilleur classement, c'est-à-dire celui qui correspond au classement des serveurs selon le nombre de documents pertinents qu'ils contiennent.
Nous effectuons cette analyse dans nos deux environnements [(TREC8.1 , Okapi), ...,(TREC8.4 , Okapi)] et [(TREC9.1 , Okapi), ...,(TREC9.8 , Okapi)]. Afin d'effectuer cette évaluation, nous choisissons d'utiliser deux métriques normalisées qui sont décrites dans la section 4.3.4.

Nous avons élaboré quatre graphiques présentés dans les figures 4.1 et 4.2. Ces figures représentent, respectivement, les variations de P_p dans TREC8 et TREC9 et les variations de R_p dans TREC8 et TREC9 en fonction de p. Les valeurs de P_p et R_p présentées dans les graphiques sont en réalité les moyennes, respectivement, de P_p et R_p obtenues par les 50 requêtes correspondantes à chaque environnement. Chaque figure présente cinq courbes correspondant chacune à l'une des approches concernées.

Les courbes de P_p

En étudiant les deux graphiques de la figure 4.1 concernant les variations de P_p, nous constatons les faits suivants :

1. les proportions des documents pertinents retournées par les serveurs sélectionnés selon nos deux approches et *CORI* est plus proche de la proportion optimale dans l'environnement TREC8 que dans l'environnement TREC9. Ceci est dû au nombre de serveurs dans chacun des environnements, en effet, le risque d'erreur dans la sélection est moindre lorsque le nombre de serveurs diminue ;

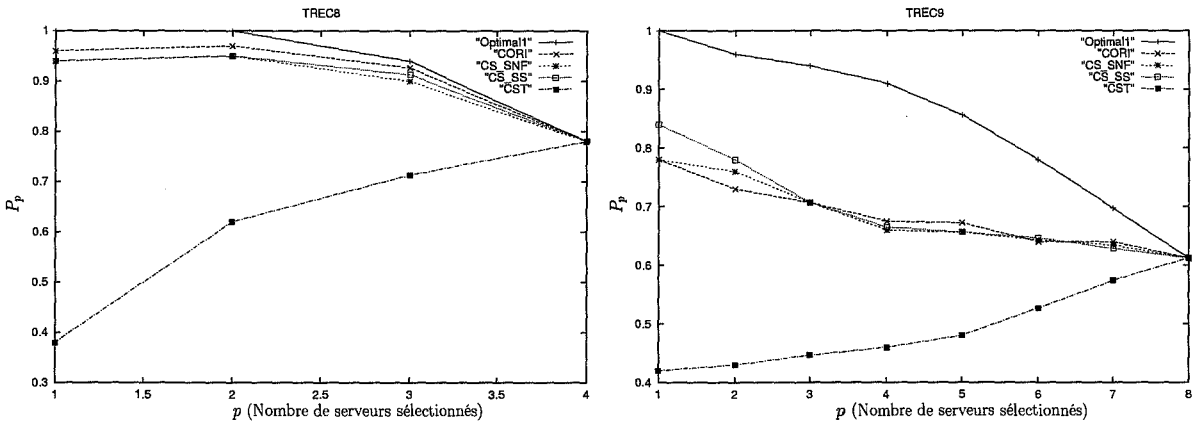


FIG. 4.1 – *Variations de P_p en fonction du nombre de serveurs sélectionnés dans les environnements* TREC8 *(à gauche) et* TREC9 *(à droite).*

2. les trois approches *CORI*, *CS-SNF* et *CS-SS* effectuent des classements plus adaptés que le classement constant et ceci dans les deux environnements ;

3. dans l'environnement de TREC8, nous constatons une légère hausse de P_p atteinte par *CORI* par rapport à nos deux approches. Cette hausse indique que *CS-SNF* et *CS-SS* classent, pour certaines requêtes, des serveurs non pertinents à différents rangs. Après étude détaillée des résultats requête par requête, nous avons constaté que, pour 47 requêtes parmi les 50, *CS-SS* et *CS-SNF* atteignent des valeurs de P_p supérieures à celles de *CORI*. Néanmoins les quelques cas d'erreur de classement surviennent au premier rang. En d'autres termes, pour trois requêtes, *CS-SNF* et *CS-SS* classent un serveur qui ne contient aucun document pertinent au premier rang ;

4. dans l'environnement de TREC9, les courbes de *CORI*, *CS-SNF* et *CS-SS* s'entremêlent, avec un léger avantage pour nos approches par rapport à *CORI* lorsque un, deux ou trois serveurs sont sélectionnés. *CORI* par contre propose un meilleur classement lorsque quatre ou cinq serveurs sont sélectionnés.

En conclusion, d'une manière globale les trois approches *CORI*, *CS-SNF* et *CS-SS* ont le même niveau de performance, mesuré par P_p, dans la détection des serveurs non pertinents. Lorsque le nombre de serveurs est plus restreint, le classement effectué par ces approches s'approche plus du classement optimal.

Les courbes de R_p

En étudiant les deux graphiques de la figure 4.2 concernant les variations de R_p, nous constatons que les courbes de nos approches se recouvrent avec celles de *CORI*. Dans les deux environnements TREC8 et TREC9, nos approches permettent de retrouver une proportion de documents pertinents légèrement supérieure à celle de *CORI*.

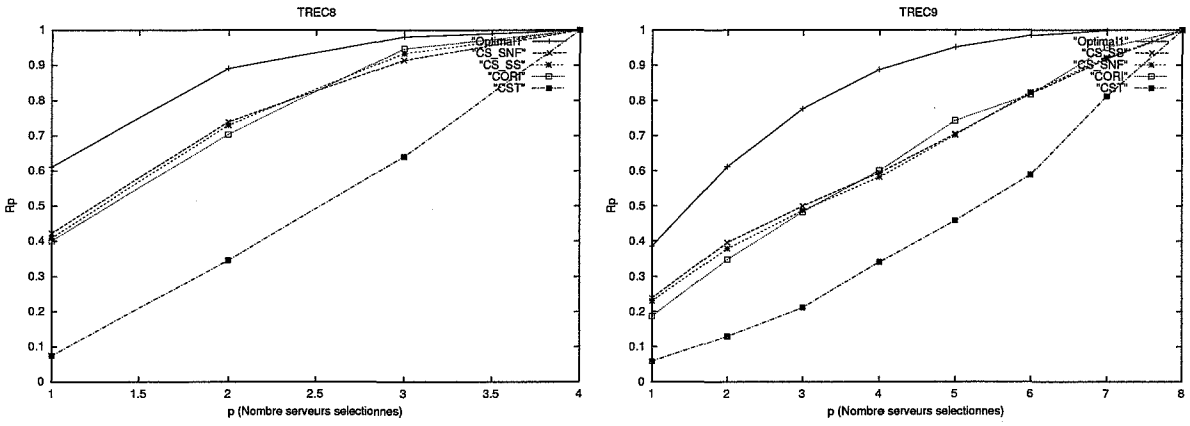


FIG. 4.2 – *Variations de R_p en fonction du nombre de serveurs sélectionnés dans les environnements* TREC8 *(à gauche) et* TREC9 *(à droite).*

Il ressort de cette comparaison , qu'en général, nos approches classent aussi bien (voire un peu mieux) les serveurs que l'approche *CORI*. Ces trois approches n'atteignent pas le classement optimal.
La légère hausse de performance de nos approches par rapport à la performance de *CORI* explique en partie la différence de précision moyenne. Néanmoins, nous pensons que l'étape de sélection de serveurs qui survient après le classement de ces derniers s'avère aussi importante. En effet, comme nous pouvons le voir dans l'exemple qui suit, le nombre de serveurs sélectionnés est important.

La figure 4.3 illustre les classements de serveurs efféctués par les approches *CORI* et *CS-SS* pour la $28^{ème}$ requête-test du corpus TREC8. Ces classements ainsi que le nombre de documents pertinents contenus dans chaque serveurs classé sont également indiqués dans cette figure.

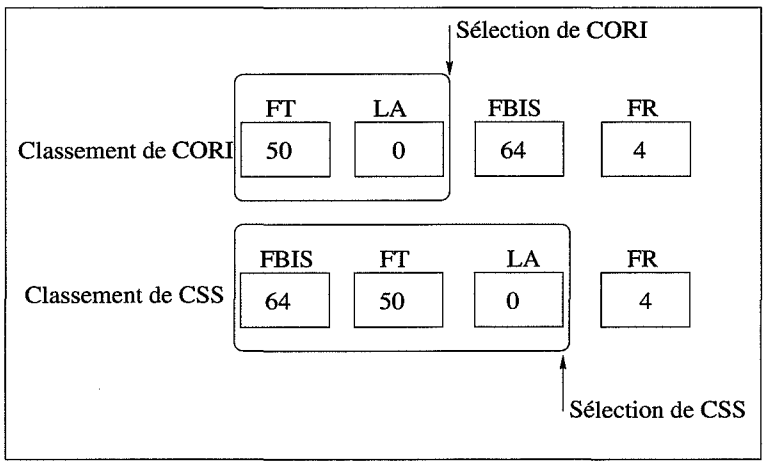


FIG. 4.3 – *Exemple de classement et de sélection effectués par CORI et CS-SS, sur la $28^{ème}$ requête-test de* TREC8*, où nous constatons l'importance du nombre de serveurs sélectionnés*

Cet exemple démontre que 64 documents pertinents sont perdus dans le cas de *CORI* car la sélection n'a pas inclue le $3^{ème}$ serveur.

4.6 Conclusion

Nous avons présenté dans ce chapitre les différentes expérimentations que nous avons conduites afin de vérifier la validité et la performance des approches de sélection de serveurs que nous avons proposées.

Ce chapitre était composé de deux parties. Dans la première partie, nous avons commencé par introduire les collections-tests, TREC8 et TREC9, utilisées dans la partie expérimentale. Nous avons ensuite présenté la construction des environnements distribués à partir de ces collections. Nous avons détaillé aussi les approches (l'approche centralisée et les approches de sélection de serveurs) que nous avons testées et comparées, ainsi que les méthodes de fusion des résultats et les méthodes d'évaluation des SRIDs que nous avons employées. Dans la deuxième partie de ce chapitre, nous avons décrit et commenté les expérimentations effectuées ainsi que leurs résultats.

Pour évaluer les trois approches de sélection de serveurs proposées dans cette thèse, nous avons d'abord cherché expérimentalement les valeurs optimales des paramètres sous-jacents. Nous avons ensuite comparé la précision moyenne de nos approches avec celle de l'approche centralisée et celle de *CORI*.
Les conclusions principales que nous tirons de cette étude expérimentale sont les suivantes :

- Il existe des approches de sélection qui permettent aux SRIDs d'atteindre la performance des approches centralisées.
- Lorsque l'environnement distribué est composé de serveurs ayant des performances différentes, nos approches de sélection (*CS-SNF* et *CS-SS*) détectent les serveurs les moins performants.
- Les résultats de la recherche distribuée utilisant nos deux approches de sélection (*CS-SNF* et *CS-SS*) atteignent la performance de l'approche centralisée lorsque nous utilisons la mesure de la précision moyenne. En outre, les résultats de *CS-SNF* sont nettement plus satisfaisants par rapport à l'approche centralisée lorsque nous étudions la précision après quelques documents retournés à l'utilisateur.
- Comparées à *CORI*, nos approches sont nettement supérieures lorsque le nombre de serveurs du SRID est réduit (cas de TREC8). Toutefois, dans l'environnement TREC9, nos approches ne s'avèrent pas significativement supérieures à *CORI*. Cependant et contrairement à *CORI*, nos approches ne nécessitent pas de coopération des serveurs, d'une part, et ne nécessitent pas de maintenance vu qu'elles ne sauvegardent aucune information concernant les contenus des serveurs.

Chapitre 5

Conclusion et perspectives

Table des matières

5.1 Contexte de la thèse

Le thème de cette thèse concerne le domaine de la recherche d'information distribuée (RID). Un système de recherche d'information distribuée (SRID) gère la recherche sur un ensemble de collections de documents distribuées soit sur un réseau local, soit sur un réseau plus étendu. Un SRID, dans sa configuration la plus simple, se compose d'un courtier et d'un ensemble de serveurs. Chaque serveur détient une collection de documents et un système de recherche d'information (SRI) qui assure la recherche dans cette collection. Le courtier représente la composante avec laquelle l'utilisateur communique. À la réception d'une requête, le courtier choisit un sous-ensemble de serveurs parmi ceux qu'il connaît, auxquels il achemine la requête. Cette opération est appelée *sélection de serveurs* ou encore *sélection de collections.* Chaque serveur ainsi sélectionné traite la requête et renvoie une liste de réponses au courtier. Ce dernier se charge alors de construire, à partir des différentes listes reçues, une liste globale qui sera retournée à l'utilisateur. Cette opération est appelée *fusion des résultats.*

La performance d'un SRID dépend, dans un premier lieu, de la performance de l'opération de sélection de serveurs. En effet, le résultat d'une recherche à l'aide d'un SRID contiendra moins de *bruit* si la méthode de sélection utilisée évite l'interrogation des serveurs dont les résultats comportent des documents non pertinents. De plus, il y aura moins de *silence* dans la liste des résultats

retournée par un SRID si les serveurs contenant des documents pertinents sont sélectionnés.

Un autre facteur qui contribue à la performance d'un SRID est l'opération de fusion de résultats. En effet, l'utilisateur sera d'autant plus satisfait si les documents les plus pertinents sont classés en tête de la liste de réponse.

Il paraît donc important de concevoir des méthodes performantes de sélection de serveurs et de fusion des résultats afin d'arriver à mettre en place des SRIDs satisfaisants.

5.2 Problème abordé

Dans cette thèse, nous nous sommes intéressés à la conception et à l'évaluation d'approches de sélection de serveurs.

Pour effectuer le choix des serveurs à interroger pour une requête donnée, le courtier prédit souvent l'utilité de chaque serveur en fonction de la requête. L'utilité d'un serveur, pour une requête donnée, se traduit par sa capacité à enrichir la liste finale par des documents pertinents.

Afin de prédire l'utilité d'un serveur, la majorité des méthodes de sélection utilisent des informations décrivant le contenu des collections. La nature de ces informations diffère d'une méthode à l'autre mais, habituellement, ce sont des données statistiques sur les termes contenus dans les collections, ces données sont appelées représentants des serveurs. Le courtier acquiert ces représentants auprès des serveurs, et se charge de les actualiser lors de changements intervenant au niveau des collections. L'acquisition des représentants des serveurs ainsi que leur mise à jour se réalise par une coopération entre les serveurs et le courtier.

Nous avons abordé le problème de sélection de serveurs selon deux points de vue à savoir, l'applicabilité et la performance. Une méthode de sélection est applicable si son implémentation dans un environnement réel s'avère facile. Elle est performante si elle permet d'obtenir des résultats satisfaisants. Dans cette optique, nous considérons peu applicable une méthode nécessitant la coopération des serveurs, ou une méthode exigeant des mises à jours régulières des représentants des serveurs. D'autre part, nous considérons, comme il est d'usage dans la RID, qu'un SRID est performant si sa performance égale celle d'un SRIC (système de recherche d'information centralisé). Ainsi, une méthode de sélection est performante si elle permet au SRID qui l'utilise d'atteindre la performance d'un SRIC.

Dans le but de concevoir une méthode de sélection applicable et performante, nous pensons qu'il est important de répondre aux questions suivantes :

- comment acquérir les représentants des serveurs sans avoir recours à la

coopération de ces derniers?

- comment choisir des représentants nécessitant peu de mises à jour?
- quelles sont les informations opportunes permettant de prédire de manière fiable la pertinence d'un serveur?

5.3 Contributions

Notre contribution à travers cette thèse se scinde en trois volets :

1. le premier volet consiste dans la conception d'une méthode d'acquisition des représentants des serveurs n'exigeant aucune coopération de ces derniers. Ces représentants sont, en outre, construits au moment de l'interrogation et ne sont sauvegardés que le temps de cette interrogation. De ce fait, aucune mise à jour de ces représentants n'est nécessaire.

   Cette méthode consiste donc à construire les représentants des serveurs pour chaque requête soumise au système. Pour ce faire, la requête est envoyée à tous les serveurs connus par le système sans exception. Ces derniers traitent la requête et renvoient une liste de résultats. Les premiers documents de chaque liste sont alors chargés et analysés au niveau du courtier. Un degré de pertinence à la requête est attribué à chaque document. L'ensemble des degrés de pertinence (calculés par le courtier) des documents retournés par un serveur constituent son représentant.

2. le deuxième concerne la manière d'utiliser les représentants des serveurs afin de prédire leur utilité et ainsi d'effectuer la sélection. En général, l'utilité d'un serveur est mesurée par un score. Nous proposons plusieurs façons de calculer le score d'un serveur en partant de son représentant. Enfin, pour effectuer la sélection des serveurs, trois principales techniques connues dans la RID sont utilisées. La première consiste à classer les serveurs selon leur score et d'en sélectionner un nombre prédéfini. La deuxième sélectionne les serveurs dont le score dépasse un seuil prédéfini. Enfin, la dernière technique calcule, en fonction du score d'un serveur, le nombre de documents qu'il doit retourner ;

3. enfin, nous avons effectué une étude expérimentale visant à mesurer la performance de nos propositions. D'une manière générale, cette étude a montré que deux de nos trois approches de sélection peuvent atteindre les résultats de l'approche centralisée et égaler voire dépasser les résultats de *CORI* (l'approche la plus connue dans les études expérimentales publiées dans le domaine de la RID). Nos approches, en plus d'être applicables, sont donc performantes.

La particularité essentielle de notre travail de thèse se résume dans le sens attribué à l'**utilité** d'un serveur. Contrairement à la majorité des études publiées

définissant un serveur utile comme étant un serveur ayant une forte probabilité de **contenir** des documents pertinents, nous considérons un serveur utile comme étant un serveur ayant une forte probabilité de **retourner** des documents pertinents. En effet, il est inutile d'interroger un serveur contenant des documents pertinents mais étant incapable de les retrouver. Cette idée a guidé notre choix d'analyser les premiers documents retournés par un serveur. Cette technique possède l'avantage de nous informer sur la capacité d'un serveur à retourner des documents pertinents car si un serveur utilise un SRI performant et contient des documents pertinents alors logiquement ces documents devraient être classés parmi les premiers documents retournés par ce serveur.

5.4 Limites

Malgré l'applicabilité de nos approches, elles possèdent néanmoins quelques limites.

D'abord, le chargement des premiers documents retournés par chaque serveur et leur analyse peut s'avérer peu efficiente. En effet, si le nombre de serveurs est considérable, ce chargement pourrait nécessiter un temps non négligeable pénalisant la rapidité du système.

Une deuxième limite touche à la disponibilité des documents. En effet, le courtier analyse les premiers documents retournés par chaque serveur afin de prédire son utilité. Or ceci s'avère impossible si les documents ne peuvent être chargés au niveau du courtier, parce qu'ils appartiennent à une organisation refusant de les fournir.

5.5 Perspectives

Deux types de directions peuvent êtres envisagées pour des travaux ultérieurs en continuité de ceux présentés dans cette thèse.

- Du point de vue théorique, notre approche d'acquisition des représentants des serveurs peut être améliorée afin de permettre une augmentation d'efficience et de performance du système. Il convient donc de chercher à réduire le temps nécessaire à la construction des représentants des serveurs en réduisant l'analyse des documents à l'analyse des résumés de ceux-ci. Si ces résumés sont disponibles dans les listes de réponse des serveurs, un gain de temps non négligeable est possible. Néanmoins, ces résumés ne sont pas toujours disponibles et surtout, ils ne sont mis à jour que si l'index du serveur l'a été.

  L'amélioration de la méthode d'analyse des premiers documents que les serveurs retournent peut s'avérer également avantageuse afin d'augmenter la performance de la sélection. Cette amélioration peut concerner la

fonction de calcul du score d'un document en introduisant de nouveaux critères tels que, par exemple, la proximité entre tous les termes de la requête dans le document ou la position des termes dans les différentes unités logiques (titre, section, etc.). On pourrait aussi envisager de calculer la proximité entre tous les documents chargés, de les regrouper et, enfin, de sélectionner les serveurs dont les documents retournés sont sémantiquement proches. On pourrait aussi examiner d'autres fonctions de calcul du score d'un serveur afin d'améliorer la sélection.

Une autre perspective possible est d'étendre le champ d'application du SRID à différents types de documents. L'extension peut porter aussi sur le domaine du multilinguisme. En effet, il est intéressant qu'un SRID puisse interroger des serveurs possédant des collections rédigées dans des langues différentes et de pouvoir retourner à un utilisateur des documents écrits dans diverses langues. Cette extension nécessite l'introduction d'une procédure de traduction dans le courtier afin de pouvoir traduire les requêtes des utilisateurs et d'adapter la procédure d'analyse des documents aux différentes langues.

Enfin, il convient de concevoir une méthode de fusion utilisant les informations contenues dans les représentants des serveurs afin de pondérer les résultats de ces derniers, ou simplement utiliser les scores des serveurs calculés par nos approches pour cette pondération.

- Du point de vue expérimental, plusieurs perspectives sont à explorer.

  Nos tests ont été effectués sur deux environnements où les résultats obtenus sont encourageants. Le premier environnement contient quatre serveurs, renfermant chacun des documents provenant de la même source. Le deuxième, contient huit serveurs de taille approximativement identiques. Nos approches, n'étant pas dédiées à un environnement particulier, il serait intéressant de vérifier leur validité dans d'autres environnements comme, par exemple, des environnements :

  a) contenant un plus grand nombre de serveurs ;
  b) où les collections sont thématiques ;
  c) où les collections ont des tailles très différentes.

  Il serait aussi intéressant de tester nos approches avec un plus grand nombre de requêtes.

  Dans nos tests nous avons varié les paramètres de nos approches afin d'obtenir une configuration optimale de ces derniers. Nous pensons qu'il serait opportun d'étudier l'impact de chacun des paramètres sur les résultats du système.

  Enfin, nous pensons qu'une implémentation de nos approches dans un environnement réel, par exemple, comme un méta-moteur sur le Web, serait une excellente validation de nos propositions. En effet, cette implémentation permettrait de découvrir le comportement de nos approches en terme d'efficience et de performance dans un environnement réel.

Annexe A

Stratégies de pondération

Pour décrire un modèle de recherche d'information, le programme Smart [Sal89] convient de représenter les documents et les requêtes par des vecteurs de termes pondérés. L'appariement est le produit scalaire entre les deux vecteurs représentant respectivement le document et la requête, selon la proposition de Salton [Sal89]. Smart représente la fonction de pondération par trois lettres codant chacune une fonction. La première lettre code une fonction de la fréquence de t dans d (ou dans q). La deuxième code une fonction de la fréquence de t dans le corpus. Enfin, la troisième lettre code une fonction de normalisation. Le tableau suivant récapitule les différents codes et leur formules correspondantes.

n	new_tf = tf_{ij} fréquence d'occurence de T_j dans le document D_i
b	new_tf = poids binaire (0 ou 1)
a	new_tf = $0.5 + 0.5 \cdot (tf_{ij}/max\ tf\ dans\ D - i)$
l	new_tf = $ln((tf_{ij}) + 1.0$
l	new_tf = $(ln((tf_{ij}) + 1.0)/(1.0 + ln(moy(tf\ dans\ D_i)))$
n	new_wt = new_tf (aucune conversion)
t	new_wt = new_tf$\cdot ln(N/df_j)$
p	new_wt = new_tf$\cdot ln((N - df_j)/df_j)$
n	w_{ij} = new_wt (aucune conversion)
c	division de chaque new_wt par $\sqrt{(\sum new\_wt^2)}$ pour obtenir w_{ij}
u	$w_{ij} = new\_wt/((1-c) \cdot moy(nt) + c \cdot nt_i)$

Annexe B

Exemple de document

Le document WTX008-B37-10 pris dans la collection-test TREC9 :

```
<DOC>
<DOCNO>
WTX008-B37-10
<TEXT>
<TITLE>
Cattery Lopend Vuur
<TEXT>
<H1>
Welcome in our cattery
<TEXT>
<A HREF="INFO.HTM">
1.The ten most frequently asked questions about the Bengal
<TEXT>
"Under construction"
<A HREF="CATWALK/CATWALK.HTM">
 2. The Bengal Catwalk
<TEXT>
 with Pictures of our cats and maybe in future of other bengals as well
3.our booklet about the Bengal
<A HREF="GOALS.HTM">
The Bengal a unique breed
<TEXT>
<A HREF="BREEDING.HTM">
4.Judging and breeding Bengal cats

[...]

<H3>
any comments or questions send us a small note
<A HREF="MAILTO:BENGAAL@XS4ALL.NL">
send E-mail
<TEXT>
<TEXT>
<DOC>
```

Annexe C

Exemple de requête

Requête 468 prise de la collection-test TREC9 :

<num>Number: 468

<title>incandescent light bulb

<desc>Description:
Find documents that address the history of the incandescent light bulb.

<narr>Narrative:
A relevant document must provide information on who worked on the development of the incandescent light bulb. Relevant documents should include locations and dates of the development efforts. Documents that discuss unsuccessful development attempts and non-commercial use of incandescent light bulbs are considered relevant.

Annexe D

Exemple de jugements de pertinence

Parties des jugements de pertinence, associés à la collection-test TREC9, pour les requêtes 474 et 475 :

```
[...]
474 0 WTX048-B50-157 1
474 0 WTX051-B10-399 1
474 0 WTX055-B13-278 1
474 0 WTX055-B13-301 1
474 0 WTX055-B16-362 1
474 0 WTX058-B32-275 1
474 0 WTX060-B28-61 1
474 0 WTX064-B29-63 1
474 0 WTX064-B35-159 1
474 0 WTX065-B10-274 1
474 0 WTX065-B20-8 1
474 0 WTX067-B38-371 1
474 0 WTX069-B20-77 1
474 0 WTX071-B29-23 1
474 0 WTX072-B23-156 1
474 0 WTX076-B29-284 1
474 0 WTX078-B39-247 1

[...]

475 0 WTX023-B35-140 2
475 0 WTX025-B12-255 1
475 0 WTX025-B18-61 1
475 0 WTX033-B21-242 1
475 0 WTX033-B46-531 2
475 0 WTX060-B03-111 1
475 0 WTX060-B11-8 1
475 0 WTX060-B12-143 1
475 0 WTX060-B12-97 1
475 0 WTX060-B13-80 1
[...]
```

Bibliographie

[Bau99a] Christoph Baumgarten. *Probabilistic Information Retrieval in a Distributed Heteregeneous Environment.* PhD thesis, XX, 1999.

[Bau99b] Christoph Baumgarten. A probabilistic solution to selection and fusion problem in distributed information retrieval. In *the Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval Berkley, CA USA*, 1999.

[BAZ+97] S. Bhattacharjee, M.H. Ammar, E.W. Zegura, V. Shah, and Z. Fei. Application-layer anycasting. In *INFOCOM 97*, 1997.

[BY99] Ricardo Beza-Yates. *Modern Information Retrieval.* Addison Wesley, 1999.

[CBH00] N. Craswell, P. Bailey, and D. Hawking. Server selection in the world wide web. In *The Fifth ACM Conference on Digital Libraries*, pages 37–46, 2000.

[CC96] R.L. Carter and M.E Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report BU-CS-96-007, Boston University, march 1996.

[CCB95] C.L.A. Clarke, G.V. Cormack, and F.J. Burkowski. Shortest substring ranking (multitext experiments for trec-4). *TREC4 NIST Publication*, pages 295–304, 1995.

[CCD99] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *ACM/SIGMOD*, Philadelphia, USA, 1999.

[CH95] A.S. Chakravarthy and K.B. Haase. Netserf: Using semantic knowledge to find internet information archives. In *The 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, Seattle, WA, July 1995.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *The Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval , Seattle, WA*, July 1995.

[Cra00] N.E. Craswell. *Methods for Distributed Information Retrieval.* PhD thesis, Australian National University, 2000.

[DAEA96] R. Dolin, D. Agrawal, and A. El-Abbadi. Classifying network architectures for locating information sources. Technical Report TRCS96-23, Department of Computer Science, University of California, Santa Barbara, september 1996.

[DH97] D. Dreilinger and A.E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 3(15):195–222, July 1997.

[DKMSZ98] Owen De Krester, Alistair Moffat, Tim Shimmin, and Justin Zobel. Methodologies for distributed information retrieval. In *The 18th International Conference on Distributed Computing Systems*, 1998.

[FG99] Y. Fan and S. Gauch. Adaptative agents for information gathering from multiple, distributed information sources. In *AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University*, 1999.

[Fou] A National Science Foundation. A national science foundation project. http://www.ucdavis.edu/whois-plus.

[FPC+99] James C. French, Allison L. Powell, Jamie Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the performance of database selection algorithms. In *The Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Berkley, CA USA, 1999.

[Fuh99] Norbert Fuhr. A decision-theoretic approach to database selection in networked ir. *ACM Transactions on Information Systems*, 17(03):229–249, July 1999.

[GGM95] L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In *21st VLDB Conference*, pages 78–89, 1995.

[GGMA99] Luis Gravano, Hector Garcia-Molina, and Tomasic Anthony. Gloss: Text-source discovery over the internet. *ACM Transactions on Information Systems*, 24(03):229–264, June 1999.

[GS95] J. Guyton and M. Schwarz. Locating nearby copies of replicated internet servers. Technical Report CU-CS-762-95, University of Colorado at Boulder, 1995.

[Har96] D. Harman. Overview of the forth text retrieval conference (trec-4). In *Fourth Text Retrieval Conference (TREC-8)*, 1996.

[HT99] David Hawking and Paul Thislewaite. Methods for information server selection. *ACM Transactions on Information Systems*, 17(01):40–76, January 1999.

[HVCB99a] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the trec8 web track, 1999.

[HVCB99b] David Hawking, Ellen Voorhees, Nick Craswell, and Peter Bailey. Overview of the trec-8 web track. In *Eight Text Retrieval Conference (TREC-8)*, Gainthersburg MD, November 1999.

[ISO93] ISO. Iso 9777:1993 information and documentation- commands for interctive text searching. In *International Organization for Standardization*, Geneva, Switzerland, 1993.

[JB77] K. Sparck Jones and R.G. Bates. Research on automatic indexing 1974-1976. Technical report, Computer Laboratory, University of Cambridge, UK, 1977.

[Jon72] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieva. *Documentation*, 28(1):11–20, 1972.

[KBM94] E.D Katz, M. Butler, and R. McGrath. A scalable http server : the ncsa prototype. In *First International World Wide Web Conference*, Geneva, Switzerland, may 1994.

[KM91] B. Kahle and A. Medlar. An information system for corporate users: Wide area information servers. Technical Report TMC199, Thinking Machine Coporation, April 1991.

[Kor97] Robert R. Korphage. *Information Storage and Retrieval.* Wiley Computer Publishing, 1997.

[LC00] Leah S. Larkey and Jamie Calla. Collection selection and results merging with topically organized u.s. patents and trec data. In *International Conference on Information and Knowledge Management, ACM CIKM*, pages 282–289, 2000.

[LCC96] Zhihong Lu, James P. Callan, and W. Bruce Croft. Measures in collection ranking evaluation. Technical report, Computer Science Department, University of Massachusetts, 1996.

[LG98] Steve Lawrence and C. Lee Giles. Inquirus, the neci meta search engine. In *The seventh International World Wide Web Conference*, 1998.

[MST94] Donald Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[MZ95] Alistair Moffat and Justin Zobel. Information retrieval systems for large document collections. In *The Third Text REtrieval Conference (TREC-3)*, pages 85–94, National Institute of Standards and Technology, 1995.

[Neg79] A. Negus. Development of the euronet-diane common command language. In *Third International Online Information Meeting*, pages 95–98, 1979.

[Org93] National Information Standards Organization. Z39.50-1992 common command lamguage for online interactive information retrieval. *NISO Press, Bethesda*, 1993.

[PFC00] Allison L. Powell, James C. French, and Jamie Callan. The impact of database selection on distributes searching. In *The 23rd ACM/SIGIR International Conference on Research and Development in Information Retrieval*, Athens, Greece, 2000.

[RAS01] Yves O. Rasolofo, Faïza Abbaci, and Jacques Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *Tenth International Conference on Information and Knowledge Management, ACM CIKM*, pages 282–289, 2001.

[Ris79] C.J.van Risjbergen. *Information Retrieval.* Butterworths, 1979.

[RL98] Michael Roszkowski and Christopher Lukas. A distributed architecture for resource discovery using metadata. *D-Lib Magazine*, june 1998.

[RMF97] Miguel Rio, Joaquim Macedo, and Vasco Freitas. A distributed weighted centroid-based indexing system. In *The Eighteenth Joint European Networking Conference (JENC8)*, Edinburgh, Scotland, May 1997.

[RNB98] Berthier A. Ribeiro Neto and Ramurti A. Barbosa. Query performance for tightly coupled distributed digital libraries. In *ACM/Digital Libraries Conference*, Pittsburg, PA USA, 1998.

[RW95] S.E. Robertson and M. M. Walker, S. and. Hancock-Beaulieu. Large test collection experiments on an operational interactive system : Okapi at trec. In *Information Processing and Managment*, volume 31, pages 345–360, 1995.

[Sal89] Gerard Salton. *Automatic text Processing: The transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley, 1989.

[SE95] E. Selberg and O. Etzioni. Multi-service search and comparison using the metacrawler. In *The World Wide Web Conference*, 1995.

[SR00] Jacques Savoy and Yves Rasolofo. Recherche d'information dans un environnement distribué. In *Traitement Automatique du Langage Naturel*, Lausanne, October 2000.

[SWJT01] A. Spink, D. Wolfram, B. Jansen, and Saracevic T. Searching the web: The public and their queries. *????*, 2001.

[VF95] Charles L. Viles and James C. French. Dissemination of collection wide information in a distributed information retrieval system. In *The Eighteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Seattle, WA, 1995.

[VH00] E. Voorhees and D. Harman. Overview of the Ninth Text REtrieval Conference (TREC-9), 2000.

[Voo95] Ellen M. Voorhees. Siemens trec-4 report: Further experiments with database merging. In *The fourth Text REtrieval Conference (TREC)*, Gainthersburg, Maryland, 1995.

[VT97] E.M. Voorhees and R.M. Tong. Multiple search engines in database merging. In *2nd ACM international conference on digital libraries, New York*, pages 93–102, 1997.

[WSJS01] D. Wolfram, A. Spink, B. Jansen, and T. Saracevic. Vox populi: The public searching of the web. *Journal of the American Society for Information Science and Technology*, pages 1073–1074, 2001.

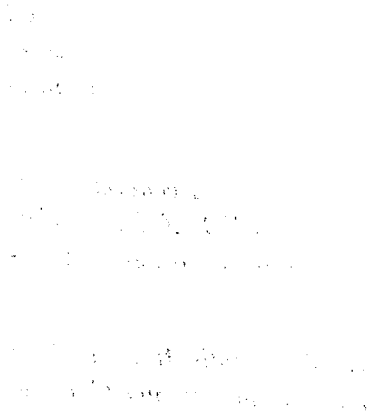
[XC96] J. Xu and B. Croft. Query expansion using local and global document analysis. In *The Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

[XC98] Jinxi XU and Jamie Callan. Effective retrieval with distributed collections. In *The Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, 1998.

[XC99] Jinxi XU and W. Bruce Croft. Cluster-based language model for distributed retrieval. In *The Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Berkley,CA USA, 1999.

[YL97] Budi Yumono and Dik L. Lee. Server ranking for distributed text retrieval systems on the internet. In *The fifth International Confernece on Database Systems for Advanced Applications*, Melbourne, Australia, April 1997.

[Zob97] Justin Zobel. Collection selection via lexicon inspection. In *The second Australian Document Computing Symposium, Melbourne, Australia*, 1997.



Bibliographie personnelle

- **F. Abbaci, J. Savoy, M. Beigbeder** *Méthodes pour la sélection de collections dans un environnement distribué..* In *CIDE2002, 5e Colloque International sur le Document Électronique.*, Hammamet, Tunisie, 20-23 Octobre 2002.

- **F. Abbaci, J. Savoy, M. Beigbeder** *A Methodology for Collection Selection in Heterogeneous Contexts..* In *IEEE-ITCC2002 International Conference on Information Technology: Coding and Computing.*, Las Vegas, 8-10 avril, 2002.

- **Jacques Savoy , Yves Rasolofo, Faïza Abbaci** *Fusion de collections dans les métamoteurs..* In *JADT 2002 6èmes Journées internationales d'Analyse statistique des Données Textuelles.*, St-Malo / France, 13-15 Mars 2002.

- **Yves Rasolofo, Faïza Abbaci, Jacques Savoy** *Approaches to Collection Selection and Results Merging for Distributed Information Retrieval..* In *ACM-CIKM'2001 Tenth International Conference on Information and Knowledge Management.* , Atlanta, November 2001.

- **Jacques Savoy, Yves Rasolofo, Faïza Abbaci** *Recherche d'informations dans des sources distribuées.* . In *INFORSID, XIXème Congrès Informatique des Organisations et Systèmes d'Information et de Décision*, Genève, mai 2001.

- **F.Abbaci** *Recherche d'informations distribuée: Application au W3 (poster).* In *JED'2000 Journée Ecrit et Document Spéciale Jeunes Chercheurs.*, Lyon, 3-6 juillet 2000.